SCALABLE WAVELET-BASED ACTIVE NETWORK STEPPING STONE DETECTION

THESIS

Joseph I. Gilbert, Major, USA

AFIT/GE/ENG/12-17

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/12-17

SCALABLE WAVELET-BASED ACTIVE NETWORK STEPPING STONE DETECTION

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Insitute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Joseph I. Gilbert, B.S. Eng. Physics

Major, USA

March 2012

AFIT/GE/ENG/12-17

SCALABLE WAVELET-BASED ACTIVE NETWORK STEPPING STONE DETECTION

Joseph I. Gilbert, B.S. Eng. Physics
Major, USA

Approved:

Lt Col David J. Robinson, PhD (Chairman)	Date
Major Jonathan W. Butts, PhD (Committee Member)	Date
Timothy H. Lacey, PhD (Committee Member)	Date

Abstract

Network intrusions leverage vulnerable hosts as stepping stones to penetrate deeper into a network and mask malicious actions from detection. This research focuses on a novel active watermark technique using Discrete Wavelet Transformations to mark and detect interactive network sessions. This technique is scalable, nearly invisible and resilient to multi-flow attacks. The watermark is simulated using extracted timestamps from the CAIDA 2009 dataset and replicated in a live environment.

The simulation results demonstrate that the technique accurately detects the presence of a watermark at a 5% False Positive and False Negative rate for both the extracted timestamps as well as the empirical tcplib distribution. The watermark extraction accuracy is approximately 92%.

The live experiment is implemented using the Amazon Elastic Compute Cloud (EC2) service. The client system sends marked and unmarked packets from California to Virginia using stepping stones in Tokyo, Ireland and Oregon. Five trials are conducted in which the system sends three simultaneous watermarked samples and one unmarked sample 100 times to each target. The live experiment results are similar to the simulation and provide statistical evidence demonstrating the effectiveness in a live environment to identify stepping stones.

I would like to thank God for all the blessings in my life. I dedicate this work to my wife and family. My parents always encouraged me and mustered any resources necessary to help me succeed. The endless hours, patience and love my wife shows on a daily basis kept me going through this program and her encouragement kept me striving to do my best throughout. I also must not forget my Chihuahua for his entertainment and help with statistical analysis.

Acknowledgments

I wish to thank my advisor, Lt Col Robinson for his support and direction and wish him a safe return from his deployment. I thank Major Butts for the additional direction and insightful comments during Lt Col Robinson's absence. As the final committee member, I also thank Dr. Lacey for his generous input and for his work as an AFIT instructor. All the AFIT instructors provide invaluable education and truly are of the highest caliber.

I also with to thank Amir Houmansadr for providing me a sample of his extracted datasets to compare to previous research. His datasets assisted me in validating my algorithm against already proven techniques in this field.

Finally, I also wish to acknowledge the use of the CAIDA dataset in this research. Support for CAIDA's Internet Traces is provided by the National Science Foundation, the US Department of Homeland Security, and CAIDA Members.

Joseph I. Gilbert

Table of Contents

Page

Abstract iv

Dedication v

Acknowledgments vi

List of Figures x

List of Tables xiii

List of Acronyms and Abbreviations xiv

1 Introduction and Motivation 1
- 1.1 Background 1
- 1.2 Problem Definition and Goal 4
- 1.3 Approach 4
- 1.4 Research Contributions 4
- 1.5 Assumptions and Limitations 5
- 1.6 Thesis Organization 5

2 History and Background 7
- 2.1 Why Stepping Stones are Important 7
  - 2.1.1 Evolution of Stepping Stones 7
  - 2.1.2 Botnets 9
    - 2.1.2.1 Botnet Overview 9
  - 2.1.3 Compromised Hosts and Stepping Stones 10
  - 2.1.4 Legitimate Stepping Stones 11
  - 2.1.5 Interactive Session Models 12
- 2.2 Initial Detection and Classification 12
- 2.3 Passive Detection 13
  - 2.3.1 Host-based Passive Detection 13
  - 2.3.2 Network-based Passive Detection 14
- 2.4 Active Detection 17
  - 2.4.1 Host-based Active Detection 17
  - 2.4.2 Network-based Active Detection 17
    - 2.4.2.1 Active detection based on packet payload 18
    - 2.4.2.2 Active detection using timing information 19
- 2.5 Other Detection Methods 23

2.5.1 Hybrid Detection 23
2.5.2 Data Mining 23
2.6 Error Correction and Coding 25
2.7 Evading Detection 25
2.8 Summary 27

3 Methodology 28
3.1 Problem Definition 28
3.1.1 Goals and Hypothesis 28
3.1.2 Approach 29
3.1.2.1 Analysis 29
3.1.2.2 Comparison to Previous Data 30
3.1.2.3 Inter-packet Delay of Secure Shell (SSH) Streams 31
3.1.2.4 Discrete Wavelet Transform 32
3.2 System Boundaries 33
3.3 System Services 35
3.4 Workload 37
3.5 Performance Metrics 38
3.6 System Parameters 39
3.7 Factors 40
3.8 Evaluation Technique 41
3.9 Experimental Design 43
3.10 Methodology Summary 44

4 Analysis and Results 46
4.1 Traffic Analysis 46
4.2 Algorithm Development 47
4.2.1 Frame Length 47
4.2.2 Synchronization Frame 49
4.2.3 Frame Size 49
4.2.4 Threshold 50
4.3 Watermark Application and Invisibility 50
4.4 Simulation 52
4.4.1 Pilot Studies 52
4.4.2 Threshold determination 52
4.4.3 Performance of the system 52
4.4.4 Comparative Analysis 53
4.5 Live Experiment Results 54
4.5.1 Watermark Statistic Analysis 55
4.5.2 False Positives and False Negatives 55
4.5.3 Dataset Anomaly 56
4.5.4 Watermark Bit Errors 58
4.5.5 Wilcoxon Rank Sum Test 59

4.5.6 Results of five trials . . . 61
4.6 Validation . . . 62
4.6.1 Comparing the Simulated and Live Experiments . . . 62
4.6.2 Detection Rates . . . 64
4.7 Analysis Summary . . . 65

5 Conclusions . . . 69
5.1 Research Impacts . . . 69
5.2 Attacker Defenses Against Algorithm . . . 70
5.3 Future Work . . . 71
5.3.1 SSH Traces . . . 71
5.3.2 Algorithm Improvements . . . 71

Appendix A: Client and Server Development . . . 74

Appendix B: Removed Samples . . . 78

Bibliography . . . 99

List of Figures

Figure Page

1.1 2008 origins of US cyber attack attempts by country [17] 2

1.2 System Block Diagram 3

2.1 Example of an attacker using stepping stones to conceal original location [3] . . 9

2.2 SWIRL Slot Numbering ($m = 4, r = 2$) [8] 22

2.3 Examples of Evading Detection 26

3.1 Example Plots of Extracted Timestamps and Houmansadr Data 31

3.2 Example Density Plots of Inter-packet Delays for Extracted Data and Houmansadr Data 32

3.3 Distribution of Differential Time Intervals for SSH Streams. Three values chosen from the 2009 January CAIDA dataset to show bimodal characteristics. 33

3.4 Algorithm of the Watermarker 34

3.5 Algorithm of the Detector 35

3.6 The System Under Test and Component Under Test 36

3.7 System Inputs, Parameters and Outputs. 40

3.8 Amazon Elastic Compute Cloud Locations 43

4.1 Distribution of the first three detail vectors for the four-level Haar Discrete Wavelet Transform using time differentials as inputs. 47

4.2 A graphical illustration of the watermark applied to a sequence of time packets. A sign change in the d^1 vector appears like two timestamps changed places in time. 48

4.3 Density plot illustrating the unmarked, watermarked and watermarked sample with simulated noise. 51

4.4 Density plot illustrating the estimated probability distributions of the watermark statistic. The threshold of $\gamma = 0.01043$ is also shown. 53

4.5 Density plot of Watermark Statistics over all servers. Unmarked data from all servers is grouped into unmarked values. 56
4.6 Density of Unmarked Live Experiments. 57
4.7 Density of watermarked sessions during live experiment. 58
4.8 Histogram of bit errors occurring during two trials. Only watermarked streams with bit errors present are represented in this plot. 59
4.9 Confidence Intervals for the False Positive, False Negative and Error Rates for the five trials. 63
4.10 Box Plots of False Positive and False Negative rates expressed as a percentage over five trials. Each trial represents 100 data points for unmarked streams and 300 samples for watermarked streams. 64
4.11 Box Plot of Watermark Error Rate of all watermarked data streams over five trials, 300 samples each trial. 65
4.12 Density plot of Watermark Statistics over All Servers. Unmarked data from all servers is grouped into unmarked values. 66
4.13 Density plot separating Watermarked and Unmarked Simulated and Live Datasets. 68
B.1 Plot of Erroneous Port 22 Times removed from experiment 79
B.2 Plot of Erroneous Port 22 Times removed from experiment 80
B.3 Plot of Erroneous Port 22 Times removed from experiment 81
B.4 Plot of Erroneous Port 22 Times removed from experiment 82
B.5 Plot of Erroneous Port 22 Times removed from experiment 83
B.6 Plot of Erroneous Port 22 Times removed from experiment 84
B.7 Plot of Erroneous Port 22 Times removed from experiment 85
B.8 Plot of Erroneous Port 22 Times removed from experiment 86
B.9 Plot of Erroneous Port 22 Times removed from experiment 87

B.10 Plot of Erroneous Port 22 Times removed from experiment 88
B.11 Plot of Erroneous Port 22 Times removed from experiment 89
B.12 Plot of Erroneous Port 22 Times removed from experiment 90
B.13 Plot of Erroneous Port 22 Times removed from experiment 91
B.14 Plot of Erroneous Port 22 Times removed from experiment 92
B.15 Plot of Erroneous Port 22 Times removed from experiment 93
B.16 Plot of Erroneous Port 22 Times removed from experiment 94
B.17 Plot of Erroneous Port 22 Times removed from experiment 95
B.18 Plot of Erroneous Port 22 Times removed from experiment 96
B.19 Plot of Erroneous Port 22 Times removed from experiment 97
B.20 Plot of Erroneous Port 22 Times removed from experiment 98

List of Tables

Table — Page

2.1 Classification of Existing Stepping Stone Detection Techniques 13

2.2 Classification of Active Watermark Stepping Stone Detection Techniques . . . 19

3.1 System and Workload Parameters 39

3.2 Factors for Simulated Experiment 41

3.3 Factors for Live Experiment 41

3.4 Live Experiment Hardware in Amazon Elastic Compute Cloud 42

4.1 P-values for Wilcoxon Rank Sum test between unmarked datasets 60

4.2 P-values for Wilcoxon Rank Sum test between watermarked datasets 61

4.3 Five Trial data showing 95% Confidence Intervals for the extracted watermark errors, False Positive rates and False Negative rates 62

List of Acronyms and Abbreviations

botnet	bot network
C2	command and control
CAIDA	Cooperative Association for Internet Data Analysis
CIS	Caller Identification System
CITRA	Cooperative Intrusion Traceback and Response Architecture
CUT	Component Under Test
DDoS	Distributed Denial of Service
DIDS	Distributed Intrusion Detection System
DMZ	Demilitarized Zone
DSSS	Direct Sequence Spread Spectrum
DWT	Discrete Wavelet Transform
EC2	Elastic Compute Cloud
EWMA	Estimated Weighted Moving Average
FEC	Forward Error Correction
FP	False-Positive
FN	False-Negative
GiB	gibibyte
IBW	Interval Based Watermark

ICBW	Interval Centroid Based Watermark
IDIP	Intrusion Identification and Isolation Protocol
IDS	Intrusion Detection System
IRC	Internet Relay Chat
IDIP	Intrusion Identification and Isolation Protocol
IP	Internet Protocol
KiB	kibibyte
MiB	mebibyte
ML	Maximum-Likelihood
LAN	Local Area Network
PN	Pseudo Noise
PDF	Probability Density Function
RA	Repeat-Accumulate
RAINBOW	Robust and Invisible Non-Blind Watermark
SCP	Secure Copy
SNMP	Simple Network Monitoring Protocol
SWT	Sleepy Watermark Tracing
SSH	Secure Shell
SSD	Stepping Stone Detection

SWIRL	Scalable Watermark that is Invisible and Resilient to Packet Losses
SUT	System Under Test
TP	True-Positive
TiB	Tebibyte
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VoIP	Voice over Internet Protocol

Scalable Wavelet-based Active Network Stepping Stone Detection

1 Introduction and Motivation

1.1 Background

Presently, malicious users are a serious problem in the computing industry. Computer and network attacks (i.e., data exfiltration and botnets) continue to grow at an alarming rate. Intrusion detection is a diverse area of interest focused mainly on preventing these attacks. Identifying the origin of the attacks and mapping the route is important to prevent future attempts. A 2008 report shows that most of the attack attempts on US computers originated in the US as shown in Figure 1.1 [17]. As stated by a security researcher at Secureworks,

> ”We believe these statistics are significant because it clearly shows that the United States and China have a lot of vulnerable computers that have been compromised and are being used as bots to launch cyber attacks.”

One important element is detecting computer *stepping stones*. A stepping stone is a computer that is actively used as a hop point, normally inside the targeted network. It communicates with external and internal computers using bi-directional network streams. Indeed, stepping stone detection is important because the malicious actors circumvent many of the security measures (e.g., firewalls or other network barriers). By having full access to a single host, the malicious actions appear as legitimate traffic and is difficult to detect.

Recent work by Houmansadr shows that watermarking inter-packet delays in packet streams is one of the most powerful tools in detecting stepping stones [10] [8]. The

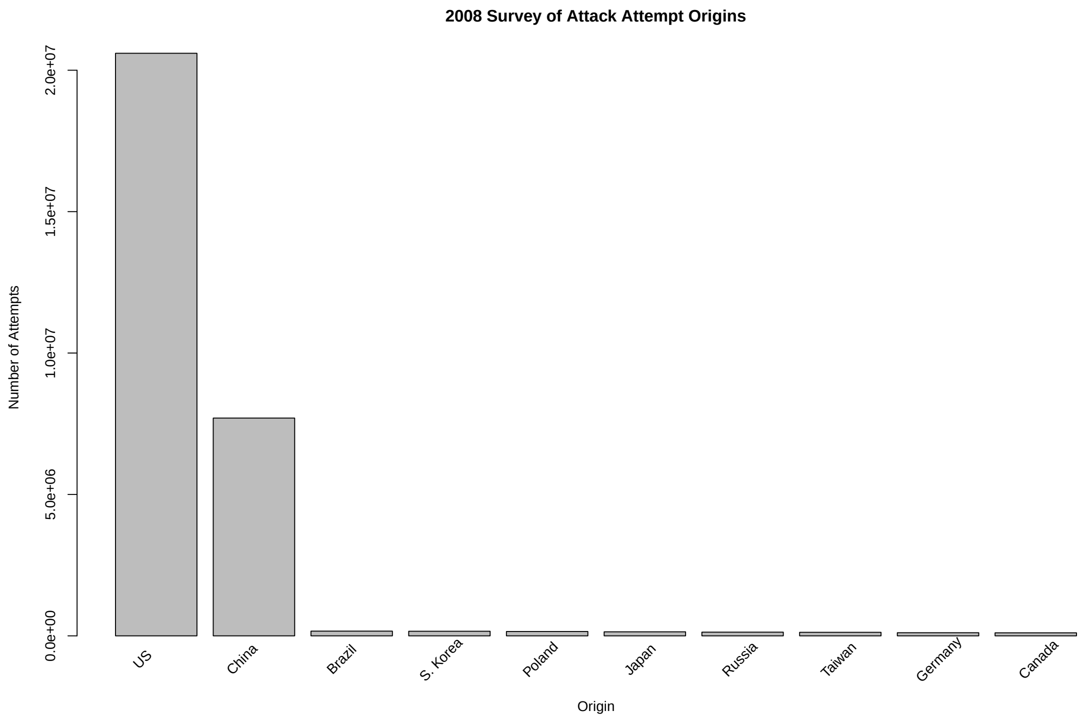


Figure 1.1: 2008 origins of US cyber attack attempts by country [17]

watermark serves as a recognizable pattern that serves as a marking for the network stepping stones. The Scalable Watermark that is Invisible and Resilient to Packet Losses (SWIRL) system watermarks the inter-packet delays to detect the network streams. This active technique adjusts the packet delays so that fewer packets are required for detection and is generally more accurate than related passive techniques. The general system model in Figure 1.2 shows the network components as well as the watermarker and detector. Placed on the edge of the network, the watermarker modifies the packet timing as they enter or leave the network boundary. If one of the network computers is used as a stepping stone, a detector placed inside the network can detect the watermarked stream from the internal network.

Previously captured streams of SSH traffic and empirical distributions included in the *tcplib* library builds the model of typical stepping stone traffic. Consistent with related research for stepping stone detection, passive data collected by the Cooperative Association for Internet Data Analysis (CAIDA) project forms the main resource for previously captured SSH streams.

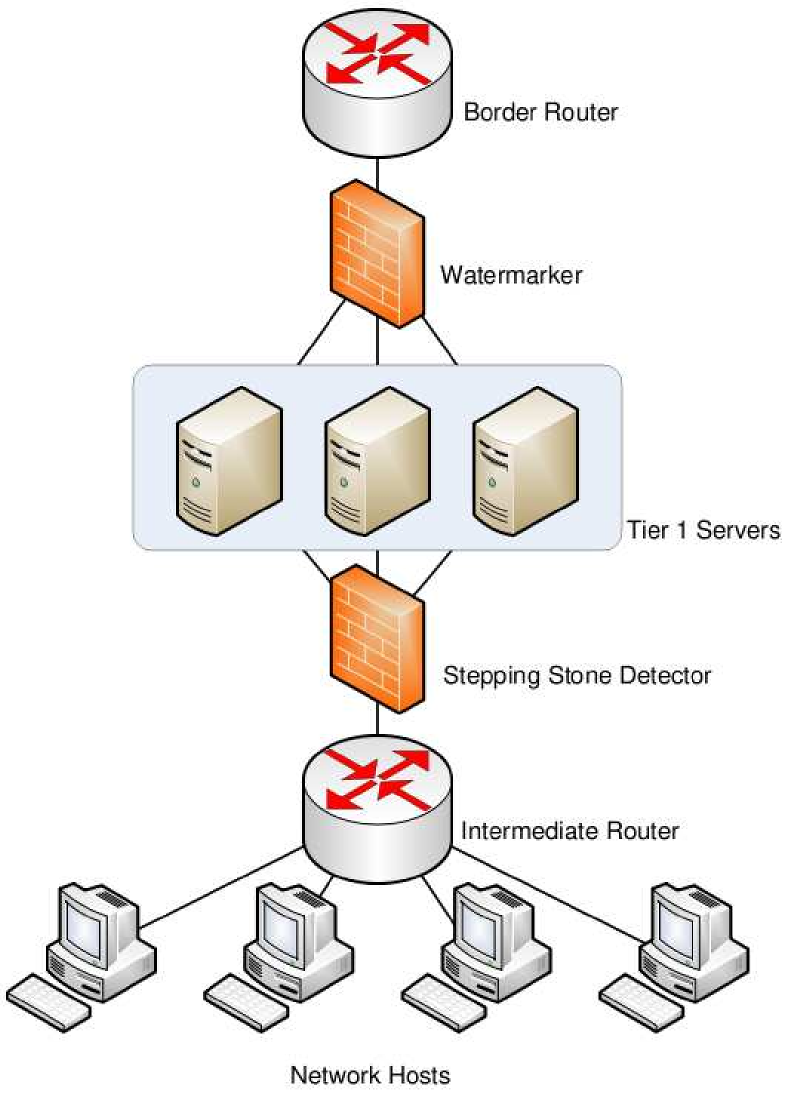


Figure 1.2: System Block Diagram

It is important to note that stepping stone detection techniques must be robust against attackers attempting to subvert detection. Attackers use chaffing as well as introduce random jitter into the packet streams to circumvent detection. Indeed, the technique must be difficult for an attacker to detect and resilient against watermark removal. The watermark must be invisible so that it is difficult for an attacker to determine if the stream is indeed watermarked. The proposed technique presents a novel method of the stepping stone detection and presents the results of the simulation and live experiment.

1.2 Problem Definition and Goal

This research introduces a novel algorithm for detecting network stepping stones. The algorithm uses a semi-blind active watermarking technique that modifies the inter-packet delays in a way that is nearly invisible (i.e., the presence of a watermark is not apparent to the attacker). The goal is to demonstrate effectiveness of identifying network stepping stones using this technique. In addition, the system must scale to facilitate integration in larger network environments. The final goal is predictable error and detection rates that are comparable to previous research.

1.3 Approach

The technique is developed based on the statistics of extracted traces from the CAIDA 2009 dataset. Next, the procedure is simulated and tested to compare performance from previous research and accepted statistical distributions. Once the algorithm is refined and evaluated, a live system using stepping stones is implemented that tests the performance in a real-world environment. A comparison between the simulation and the live experiment demonstrates the performance and effectiveness of this novel technique.

1.4 Research Contributions

This research serves to introduce and evaluate a technique to detect stepping stones in a network. If applied in a real-world environment, this system can assist network security personnel in identifying malicious network streams, combatting botnets and preventing data exfiltration.

1.5 Assumptions and Limitations

The main assumptions and limitations in this research involve three areas. The first involves the limitation that the extracted timestamps from the CAIDA 2009 SSH ports do not contain payload information and therefore are not guaranteed to be interactive sessions. These extracted timestamps are assumed to be a valid representation of typical interactive sessions.

The second area involves the selection of stepping stones for the experiment. A true valid sampling requires a random selection of true network stepping stones from the complete population. Because this is not feasible, the stepping stones selected in the experimental phase represent hosts that exhibit similar characteristics to likely stepping stones. It is assumed in this research that the results may be extended to a larger scale based on the fact that the selected stepping stones are a valid sample of the available stepping stones.

Lastly, because this research focuses on interactive stepping stones, the algorithm performance decreases when the type of traffic is changed. The performance specifically deteriorates during high rates of traffic as well as constant rate traffic. Although the algorithm could be adapted to these distributions, this research focuses mainly on the distribution associated with human driven interactive sessions. As a result, automated sessions are not within the scope of this research, however efforts to extend this technique may focus on applying the wavelet technique to these sessions.

1.6 Thesis Organization

This thesis presents the research in a manner typical of experimental research. First, the background addresses previous research relating to this field. Next, the experimental methodology and the analysis are presented. Conclusions follow the results addressing the impacts of the research and discusses future work. The appendices include a discussion on the development of the client and server programs as well as a listing of rejected CAIDA datasets.

2 History and Background

This chapter introduces many of the techniques evaluated in previous related research. Stepping stone detection traces back approximately ten years ago in 1995 when Staniford-Chen and Heberlein recognized the problem and proposed an initial solution [20]. Since then, attackers progressed in complexity making the detection of stepping stones more difficult. As the attackers' techniques became more advanced to cloak their actions, the detection methods also became more complex to adapt to these changes. Advantages and disadvantages of previous techniques are discussed in this chapter to set the stage for the proposed technique for stepping stone detection.

First, the importance of detecting the stepping stones is discussed followed by a brief discussion of malicious and legitimate uses of stepping stones. Next, a classification of the current detection techniques are presented as passive and active methods along with blind and non-blind detection. Finally, a summary of the history and previous research is presented.

2.1 Why Stepping Stones are Important

2.1.1 Evolution of Stepping Stones. The problem of intruders gaining access to computer systems through the network traces back many years and continually presents a difficult problem in today's growing networks. In 1995, Staniford-Chen and Heberlein introduced the problem of tracing an intruder back through a chain of multiple machines. These chains are often referred to as stepping stones and detecting them is commonly known as Stepping Stone Detection (SSD) [20]. The chains provide anonymity to the attacker so that tracing the original source of an attack is more difficult. Because an attacker may have access to only one host due to security or network restrictions, using a stepping stone allows the attacker to extend the reach of the attack using this one compromised machine.

Fifteen years later this same problem still exists and as the complexity of the attacker increases, this problem is more difficult to solve. Data streams are more complex due to the use of encryption, data padding or chaffing and timing perturbations. The encryption schemes hides the contents of the data from both eavesdropping or sniffing. In many cases, even the victim stepping stone cannot read the data although it traverses the link. Data padding or chaffing involves the attacker inserting extra data into the useful data stream. Chaffing is a practice in cryptography that adds packets of data that appear legitimate but are only discernible to the receiver with the correct key. The chaffing process is commonly used to maintain confidentiality without actually encrypting the data, but requires significant overhead to mask the original contents of the data. Because many applications can be identified by the inter-arrival time of the packets (e.g. most Voice over Internet Protocol (VoIP) calls), the actual packet times may be adjusted so that they cannot be easily classified. The attacker manipulates these timing perturbations to mask the actual delay of the packets. These common techniques used by attackers make the classification of the packet streams at the stepping stones more difficult to detect, though not impossible.

The importance of stepping stone research is demonstrated in government proposals. An Air Force solicitation explicitly requests research in identifying stepping stones to enable traceback [3]. Figure 2.1 illustrates an example scenario from the solicitation in which the attacker uses stepping stones to conceal its actual origin and make the traceback more difficult.

Complex bot networks (botnets) and compromised servers are some examples of present day attacks that commonly use stepping stones. These stepping stones serve as a relay point so that the attacker can disguise their tracks or penetrate deeper into a network. The botnets normally contain an overlay network for command and control which is disguised by the use of stepping stones.

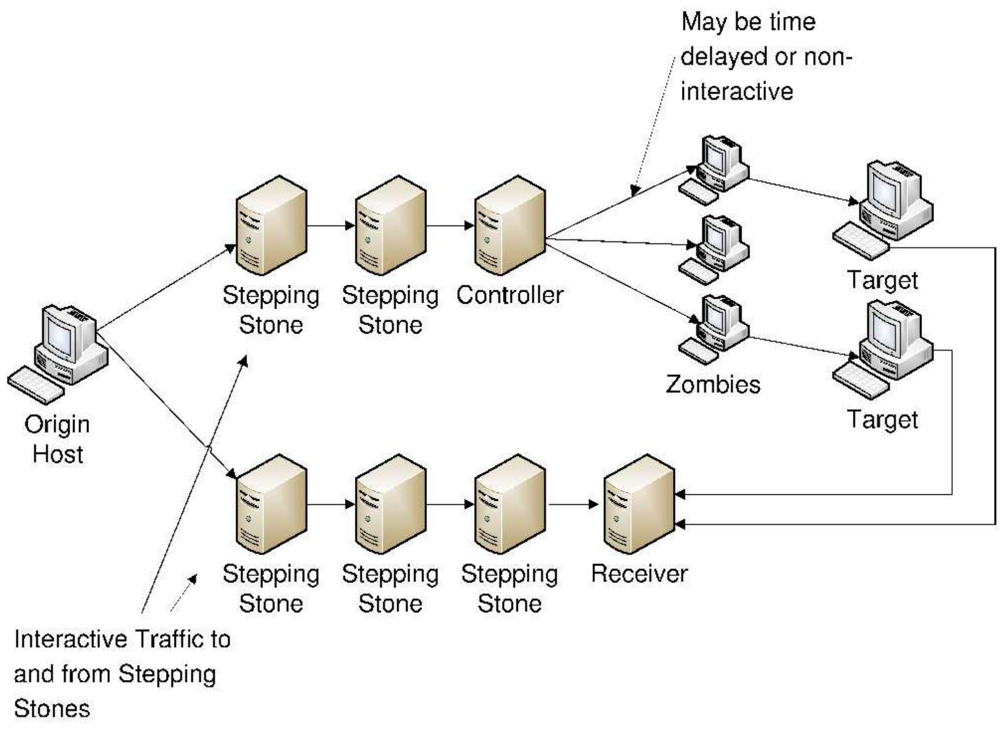


Figure 2.1: Example of an attacker using stepping stones to conceal original location [3]

2.1.2 Botnets.

2.1.2.1 Botnet Overview. A great deal of current security research focuses on analyzing and categorizing botnets. While this research does not specifically serve this purpose, a brief description of a botnet is included as it applies to the importance of the stepping stone. The botnet is generally characterized by a bot, a botmaster, and the command and control (C2) server. The bot is often the victim or end-user machine running the bot software to communicate with the C2 server. Generally, the bot receives commands issued remotely by the botmaster or botherder. Often, the commands between the botmaster and the C2 server as well as the between the C2 server and the bot are not relayed directly. Instead, stepping stones are often used to mask the origin of the C2 network. The most popular example of the stepping stone in botnets is the Internet Relay

Chat (IRC) server. In this scenario, the IRC server relays commands between the C2 host and the bots.

While the botnet software is not always malicious, the majority of security research involves preventing unknown malicious bots. Several characteristics make detecting these unknown bots difficult. First, the communication in the botnet may lie dormant for long periods of time awaiting commands. Also, botnets may be divided into smaller sections controlled by networks of C2 servers. The C2 server may direct at any time the bots to migrate between servers and divert traffic to different machines. Lastly, some machines may serve multiple botnets, creating a situation in which the amount of infected hosts appears to exceed the number of physical infected hosts. Likewise, if the C2 server only directs certain nodes to respond, the corollary may exist where the amount of infected hosts is vastly underestimated [11].

In some cases, the commands sent to the bot may not be able to traverse the endpoints in the original network configuration. In this case, stepping stones are used to gain access to a network and broaden the depth of the botnet. Strayer identifies these intermediate hosts as *rendezvous-points* and states that they may be hierarchical to support scalability in the botnet [23]. Flow analysis assists in identifying botnet traffic not originally observed. While the flow correlation does not necessarily mean a botnet is present, it does offer a more focused effort on likely targets.

2.1.3 Compromised Hosts and Stepping Stones. Besides command and control for botnets, attackers may simply use a compromised host as a stepping stone to obfuscate the origin or penetrate deeper into an internal network. As attacker techniques become more advanced, the scenarios typically include more steps. The standard Intrusion Detection Systems (IDSs) have difficulty identifying these multi-faceted attacks. Indeed, IDS enhancements include a new modeling language introduced to combat the complex scenarios [1].

Larger organizations in particular are common targets for stepping stone attacks. While the majority of networks are behind security devices (e.g., firewalls or Demilitarized Zones (DMZs)), even the compromise of one system can be used as a stepping stone to reach the rest of the computers on the network. A report on network security lists these stepping stone threats as one of the most probable avenues of attack for external facing devices [18]. These threats for embedded devices may be greater due to the difficulty in securing such devices. In fact, one resource indicates the possibility of using an embedded web server inside a picture frame as a stepping stone to access the entire company intranet [6].

2.1.4 Legitimate Stepping Stones. Not all traffic characterized as stepping stone traffic is malicious (e.g., VoIP traffic or automated polling systems). Even using decentralized peer-to-peer methodology to send this traffic may appear as stepping stones. Other uses for stepping stones are cases in which administrators only allow access to certain computers through gateways. In many cases the gateway performs like a legitimate stepping stone or hop point. Policy based decisions must also be employed in the SSD to ensure that legitimate stepping stone traffic is filtered from possible malicious traffic. However, the policy based decision must be made carefully because the malicious traffic may indeed closely resemble the legitimate traffic.

Previous research shows that VoIP traffic can be detected using stepping stone detectors even when using anonymizing services. The detector accurately detects VoIP traffic streams and identifies the endpoints by manipulating the delay between packets [26]. This research carefully modifies the delay so that the endpoints could be detected while not adversely affecting the voice quality. The research demonstrates that stepping stone detection can be extended to identify endpoints of authorized traffic. Although the VoIP traffic was authorized on the network, they were able to determine the endpoints of the conversation.

2.1.5 Interactive Session Models. Much of the stepping stone research concentrates on identifying and modeling interactive sessions based on human interactions. Initial research modeled the inter-arrival between packets as a Poisson distribution. While this distribution was the basis for many years, other distributions (e.g., the empirical *tcplib* distribution and the Pareto distribution) have proven more accurate in modeling human-driven sessions [16].

In addition to the inter-packet delays, the stepping stone research must consider other variables in the network such as jitter. A study on wide-scale WAN networks using the PlanetLab overlay network concludes that the maximum jitter is approximately normally distributed with a zero mean and a maximum standard deviation of 5 ms [14]. Additional research observes the standard deviation of the jitter between 6.2 to 12 ms [8].

2.2 Initial Detection and Classification

The initial paper by Staniford-Chen introduced the concept of using a series of computers to relay traffic [20]. By successfully logging on to each computer using a UNIX terminal, this theory established a connection to the endpoint via intermediate computers as stepping stones. The importance was illustrated by the increasing number of intrusion attempts, as far back as 1995. Research by Staniford-Chen relies on thumb printing the connections to identify the packet streams. The thumb print simply compares the data payload on either side of the host to correlate the network flows. This is easily circumvented using encryption or other data disguising technique. The concept was extended by Zhang and Paxson where they first named the network stepping stone. This work also set the stage for more robust detection methodologies including active detections [32].

The SSDs are categorized according to technique and location. The techniques are passive and active methods and the location of the SSD is host-based or network-based. Table 2.1 shows various SSD techniques [28]. Similar to other host-based models, these detection methods require sensors or adjustments made at each individual host, whereas network-based detection models take place at different points along the network path. The passive detection methods do not modify or alter the packet, whereas the active-based detection works primarily by altering the packet timing or payload.

Table 2.1: Classification of Existing Stepping Stone Detection Techniques

	Passive	Active
Host-based	DIDS, CIS	Caller ID
Network-based	Thumb printing Timing-Based Deviation-Based Online Sketching	IDIP CITRA SWT RAINBOW SWIRL

2.3 Passive Detection

In the passive SSDs, the network traffic is classified without modification. The obvious advantage is that the attacker can not determine if the detection is taking place. Acting much like a sniffer it performs operations in a covert manner because the network traffic is not modified. The passive detection can be further subdivided into host-based and network-based passive detection.

2.3.1 Host-based Passive Detection. The two main examples of host-based passive detection are the Distributed Intrusion Detection System (DIDS) and the Caller Identification System (CIS). DIDS is an early system developed by the University of California at Davis to trace events as they progressed through the network [19]. It uses event monitors and triggers to correlate network activity to the user involved. As an example, if a user “smith” logs on to another machine using a guest account, the host-based event generator triggers the Local Area Network (LAN) monitor about the new connection. From there, if the user “smith” executes a network scan using the guest account, the monitor could trace the logon attempt back to the user “smith” on the original machine. A server maintains the full login chain to trace back communications to the original user.

The CIS also uses the host-based detection approach but in a distributed manner [12]. Instead of a centralized server managing all of the connection states, when a host on machine *n* attempts to connect to another host, it presents the current login chain of *n* hosts for verification. The host at ($n + 1$) then verifies the presented chain from computer 1 to *n*. The new connection is only authorized to login when the chain is fully verified. The new host is then added to the login chain for future connections. This process introduces additional overhead to the login process but does not require the central server to manage the state of the connections.

2.3.2 Network-based Passive Detection. Although the network-based passive detection shows promise in many areas, it does not show significant improvement over using standard passive intrusion detection methods. Many of the techniques and procedures are similar in that most use some form of statistical processing to match existing data (e.g., Bayesian detection or correlation). The network-based passive detection methods include thumb printing, timing-based and deviation-based. One of the main drawbacks of the network-based approach is that it must trust the sensors to provide correct data. Most of the schemes presented for both active and passive network-based detection rely on the fact that the sensor is not compromised when verifying the integrity of the data.

The thumb printing detection method first described by [7] and extended in [20] describes a method in which the packet is assigned to a “thumb print” at a central server based on the packet payload. This thumb print is analyzed and correlated with other thumb prints to determine which connections are incoming and outgoing on a host. If the thumb print matches both connections, then that host can be identified as a stepping stone for that data stream. This relies on simple communication between hosts and is easily defeated using encryption or chaff packets.

The timing-based detection methods enhances SSD and demonstrates effectiveness at detecting stepping stones, even if traffic is encrypted. The packet inter-arrival times are analyzed using network devices and correlated to determine which packets matched both incoming and outgoing streams [32]. Similar to the thumb printing, the timing-based detection methods are based on correlation, but do not rely on the similarity of the incoming and outgoing data to create a match.

Work by Donoho and Paxson identifies three key characteristics in detecting stepping stones [5]. First, they show that random swaps of inter-packet delays within a given time window provide a statistically provable method of correlating input and output streams. Their technique selects a given time window and randomly interchanges the inter-packet delays, retaining the underlying distribution. They also demonstrate that this technique is non-causal. Second, this seminal work demonstrates applying the Discrete Wavelet Transform (DWT) on vectors of inter-packet delays. The system correlates the network streams using the DWT coefficients to identify the stepping stones. This research analytically demonstrates the effectiveness using the Poisson distribution to model the inter-arrival times. As previously mentioned, the distributions for interactive sessions are not well modeled by the Poisson distribution; however, they claim the results should be similar using the Pareto or *tcplib* distributions. The simulation using the empirical *tcplib* distribution shows promising results using the DWT correlation of coefficients. Lastly, they show that these techniques may be resilient to and possibly even detect the presence of chaff packets.

Deviation based SSD operate similarly by analyzing the minimum average delay gap between the packet streams of two Transmission Control Protocol (TCP) connections. The deviation uses both the packet timing as well as the TCP sequence number. Note that the TCP payload is not considered in the calculation. One obvious drawback of the deviation-based SSD is that it is only valid for TCP connections. Other reasons for the deviation-based approach failures were caused by Unix broadcast traffic that replicated stepping stones as well as connections with very large latencies. In the case of the Unix broadcast applications, the correlated traffic matches two flows from one distinct source (e.g., from $h_1 \rightarrow h_2$ and $h_1 \rightarrow h_3$). This could be easily sorted out since it does not conform to a stepping stone which would be $h_1 \rightarrow h_2 \rightarrow h_3$. In the latter case, if a user logged on to a system from a foreign country and then returned a connection back to the country, the algorithm had difficulty detecting the stepping stone due to the large latency [30].

One of the latest approaches involves a technique called Online Sketching as described in [2]. Online sketching uses an algorithm to examine flows at the network boundary to identify stepping stone traffic. Based on data sketches, which are widely used in stream analysis, the algorithm maintains short sketches of data streams to identify the stepping stones at the network boundaries. Through experimental verification of 100 SSH flows, online sketching identifies 95% of the stepping stone data streams even with additional perturbations, chaff packets and background traffic in the data set. One main advantage of this technique was that in traditional passive correlation analysis, $O(n \times m)$ calculations are required where n and m are the ingress and egress data flows, respectively. This process of online sketching only requires $O(n + \sqrt{n \times m})$ calculations. The main disadvantages of this approach is that to be effective against a certain flow, the parameters must be selected very carefully. Because of this, it was not robust against flows that deviated from this original characteristic. In other words, if the parameters were chosen to identify a flow with approximately 2.5 packets per second, a new flow that was 10 packets per second may not be detected with the same accuracy.

2.4 Active Detection

The active detection methods for SSD require more sophistication but offer better results in general when compared to the passive methods. Because they are able to manipulate the packets or data streams, they require codes and programs to create the desired effect. This also means that they may introduce undesirable effects (e.g., as added packet delay or bottlenecks) depending on the network performance. While the computational complexity may be reduced in the active detection, the scalability may be affected due to the network resources.

2.4.1 Host-based Active Detection. Caller ID (not to be confused with CIS) is an active detection method that was proposed by Staniford-Chen and reportedly used by the Air Force [20]. It proposes that in order for an attacker to use a chain of stepping stones, each link in the chain must contain a vulnerability. Similar to a "hack-back" approach, the Caller ID system would attempt to gain access to each host in the chain to identify the path. Obvious legal and technical restrictions make this approach somewhat unfeasible, especially to identify an attacker in real-time. While an attacker can take months to establish a series of stepping stones, the Caller ID approach may not work in a timely manner, if at all.

2.4.2 Network-based Active Detection. As the primary thrust of this research, it is important to analyze the advantages and drawbacks of past network-based active detection techniques. Early thumb printing attempts led the way to more advanced timing-based as well as deviation-based detectors able to identify stepping stones, even when obfuscated by encryption and chaffing. The active detection led the way to active *watermarks* in order to reduce the computational complexity of the algorithms and improve detection rates. The fundamental block diagram for network-based watermarking is shown in Figure 1.2.

2.4.2.1 Active detection based on packet payload. The Cooperative Intrusion Traceback and Response Architecture (CITRA), based on the original Intrusion Identification and Isolation Protocol (IDIP) system, provides an infrastructure that enables IDSs, firewalls, and other network components to cooperatively trace and block network intrusions. The original IDIP design is a protocol for reporting intrusion-related events and coordinating the attack traceback. Additionally, it allows for an automated response action and reduces the network manager's workload in the event of an attack. The CITRA system extends this to multiple functional neighborhoods that communicate with each other to correlate findings. Much of the analysis focuses on the defense against a Distributed Denial of Service (DDoS) attack, as the IDSs constantly exchange information in order to locate the attacker [21]. Although this approach does not modify the packets on the network level, it is classified as an active approach because of the information exchanged between the IDSs. Wang also questions the ability of the intermediate boundary controller to identify an intrusion solely based on a hard-coded attack description [28].

The Sleepy Watermark Tracing (SWT) approach is another active network-based detection and tracing framework [29]. It is referred to as *sleepy* because it only is triggered in the event of a detected intrusion. Once an intrusion is detected, it activates the SWT to embed the watermark in the packets. The watermark is a "virtual null string" that is injected into the packet that appears null to the end user of the application. In the case of the telnet and rlogin applications, a virtual null string may consist of a series of "\b" characters. Guardian boundary devices correlate the SWT to incoming and outgoing packet streams. An important feature of this research is the response of actual hardware to the SWT. They showed that the SWT gateway latency overhead is only approximately 50 μs. Although the SWT technique shows promising results, it is ineffective if the stepping stone uses link-to-link encryption. It also requires that each host needs a single, trusted guardian gateway to correlate the packet streams.

2.4.2.2 Active detection using timing information. Modifying packet timings is an active method in which the watermark information is embedded in the packet timings instead of the actual packet data. These techniques can be further classified as blind or non-blind depending on the amount of information passed between the watermarker and the detector. In a blind system, the detector only has access to a secret key shared between it and the watermarker. In a non-blind system, additional information about the packet stream is passed to the detector, typically through an out-of-band channel. Table 2.2 shows the classification of the current watermarking techniques as blind or non-blind.

Table 2.2: Classification of Active Watermark Stepping Stone Detection Techniques

Blind	Non-Blind
SWT	RAINBOW
ICBW	C-RAINBOW
SWIRL	

Loosely based on the active SWT, Wang proposes actively modifying the packet delays for the Interval Centroid Based Watermark (ICBW) technique in order to correlate the streams of traffic. Instead of embedding the marking in the stream data, this framework adjusts the delay of the packets to encode the correlated values in the delays. This technique has better detection rates and less False-Positives (FPs) than the passive counterpart; however, the technique does not adequately scale [27]. Later research shows ICBW successfully traced active VoIP conversations through anonymizing networks [25]. Additionally, researchers later demonstrate, however, that adjusting the inter-packet delay could be subverted using multi-flow attacks.

Another similar approach is to manipulate the arrival times of the packets using preselected time intervals. This Direct Sequence Spread Spectrum (DSSS) technique is proposed to alleviate the problem of repacketization [29]. In the case of SSH and other applications, this is a natural effect and poses a challenge to the detection of stepping stones. Each flow is sliced into short fixed-length time intervals. The watermark is embedded in these slices by manipulating the packet count during the specific intervals. The research shows that using synthetically-generated SSH traffic flows with the empirical *tcplib* distribution, it achieves 100% detection rates with under 1& FP rate.

Similar to the previously mentioned techniques, a proposal of a DSSS watermarking process shows promising results by encoding a binary watermark of n bits in an interval of length T_s. Therefore, the packet length needs to be at least nT_s long in order to encode the entire watermark. Both the watermarker and the detector must agree on the parameters for the length T_s as well as the Pseudo Noise (PN) code. The n bits are based on a PN code similar to the DSSS codes used in radio signal transmissions. This allows the detector to recover the watermark by applying a high-pass filter to the received signal and subsequently passing it through a de-spreading and low-pass filter [31].

Although the packet inter-arrival modification techniques only share the key to the detector, they are all vulnerable to multi-flow attacks as described in [13]. In this attack on the watermarking scheme, it is assumed that the attacker gained control of the stepping stone and can monitor the incoming and outgoing flows. If multiple external flows are generated to this host, then the attacker can collect the timing and recover the secret watermark key. Indeed, such actions allow an attacker to both detect and remove the watermark for ICBW and Interval Based Watermark (IBW) techniques.

The Robust and Invisible Non-Blind Watermark (RAINBOW) detection system Houmansadr proposes extends aspects of the SSD to be robust against packet losses [10]. It begins by storing the timing information for a specific flow in a database where

$t_i^u | i = 1, \ldots, n+1$, and the superscript u refers to the unmarked flow entering the watermarker. Next, it delays the packet by a value w_i with values of $\pm a$ with equal probability:

$$w_i = \begin{cases} +a & \text{w.p. } \frac{1}{2} \\ -a & \text{w.p. } \frac{1}{2} \end{cases} \tag{2.1}$$

The value a represents the amount of watermarking and is chosen small enough that it is invisible to ordinary users and attackers. In order to detect the watermark in this non-blind system, the detector has the timings t_i as well as the watermark components w_i. The detector then determines if a watermark is present based on a normalized correlation scheme. In typical connections such as SSH and any TCP connection, all packets do not have a corresponding egress packet due to repacketization or other reasons (i.e., initial SYN packets, RST packets and FIN packets). The RAINBOW system incorporates selective correlation to account for packet loss. This selective correlation adds an additional matching step in order to find and remove packets that do not have a corresponding match in the other flow using sliding windows. Although the RAINBOW technique produces promising results, it lacks efficiency and is not scalable.

The original RAINBOW technique is further extended into the C-RAINBOW by adding additional error correcting into the watermark [9]. Instead of using the n_A bits for the watermark, the C-RAINBOW uses Repeat-Accumulate (RA) codes to repeat the n_A bits q times. This repetition provides error correction to account for noisy networks and better detector performance. This model also passes the information through a soft-limiter block in order to remove excess noise from the system. The C-RAINBOW is modeled using MATLAB over 10000 samples and compared to the RAINBOW results. The simulation demonstrates that the detection rate increases using the same n_A while the number of FPs decrease. The C-RAINBOW also detects smaller values of the watermark amplitude, a, which is the "jitter" created by the watermarker.

The SWIRL is an extension of the RAINBOW technique and is the first blind active-based network detection scheme that claims to be robust against packet losses, multi-flow attacks and is scalable [8]. The system selects packet intervals of length T and divides them into r subintervals. SWIRL then further subdivides each subinterval into m slots as noted in Figure 2.2. The permutation π^{r-1} modifies the number of packets in each slot according to the secret watermark key. A base interval is also selected so that the detector can identify the information. The selection of the parameters must be carefully considered depending on the traffic characteristics. The crossover point between the FP andFalse-Negative (FN) rates is calculated and sets the ideal point for choosing the parameters for SWIRL. The SWIRL is modeled using MATLAB and previously captured traffic. In addition, live experiments over the PlanetLab infrastructure test the performance. Extracted packet timings from the CAIDA 2009 January dataset are the basis for the SSH timings. The True-Positive (TP) and FP detection rates closely match the theoretical values and demonstrate that the technique may be viable in a large scale network.

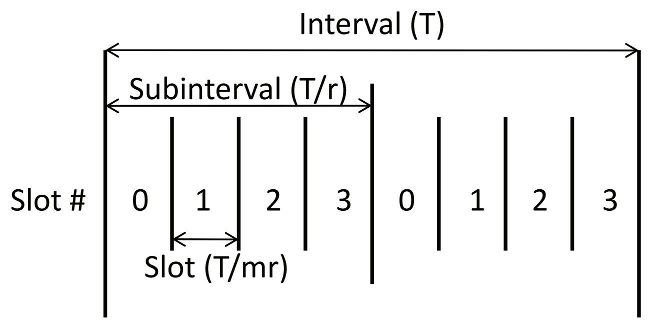


Figure 2.2: SWIRL Slot Numbering ($m = 4, r = 2$) [8]

2.5 Other Detection Methods

2.5.1 Hybrid Detection. In other works, hybrid detections propose combining active and passive techniques to form a more elaborate system for the SSD [15]. One of the hybrid approaches involve a system that integrates the host-based and network-based detection in an intelligent system. The research proposes that the idea was possible, although no metrics or proofs are derived. It is mainly a theoretical design which in theory could reduce the number of FPs and increases overall accuracy of the SSD. The main claim is that the trust problem in the host-based system could be alleviated using the data from the network-based system creating a more robust detection system. No efforts are shown to prove how effective the hybrid model would be at improving the trust in the host-based system.

2.5.2 Data Mining. Closely related to the passive correlation methods, there also is research involving data mining of packet streams [23]. Using the standard data mining techniques on a known benign botnet, one report shows that the first stage of filtering can reduce the amount of non-important data streams by a factor of 37. Beginning with 8,933,303 TCP flows, filters remove obvious packets such as large file transfers and reduce the amount of flows requiring correlation to 238,252. This is accomplished without using port classifications so that traffic masquerading on unexpected ports would still be detected. In the next classifier stage, the remaining flows are passed through three machine learning systems. Between the J48 decision tree, Naive Bayes and Bayesian Networks, the naive Bayes classifier performed best in terms of the FN rate and FP rate. The average bytes per packet also have the highest discriminatory power in detecting the botnet traffic. These two stages reduce the correlated traffic to approximately 48,000 packets from the original 8,933,303.

In the next stage, the correlation algorithm attempts to identify flows as a part of a stepping stone from the remaining packets. Instead of calculating the correlation value based on a computationally expensive algorithm of $O(n^2)$ where n is the number of active flows, they use a new technique. While the correlation could not identify all of the known botnet traffic, the outliers are attributed to implementing the software inside a virtual machine. The research concludes that the virtual machine and the reduced calculations offered by the new algorithm affects the timing preventing the necessary correlation. In the last automated stage, a topological analysis determine which of the remaining flows had common endpoints. Lastly, a manual analysis of the flows must be performed to reveal the presence of the stepping stone.

Further research described in [22] expands the improved correlation algorithm expressing the following characteristics as a time series:

- Packet event times
- Packet inter-arrival times
- Inter-burst times
- Bytes per packet
- Cumulative bytes per packet
- Bytes per burst
- Periodic throughput samples

The new algorithm examines the characteristics vector of each flow as a point in n-space. Using Euclidean distance of the points and the Estimated Weighted Moving Average (EWMA) ($\alpha = 0.75$) to calculate the moments as a running estimate, the results are then correlated. Plotting the Probability Density Function (PDF) of the distances, multiple spikes are seen corresponding to highly correlated flow pairs. In this example, they correctly hypothesized that the C2 flows were the pairs with distances closest to 0.

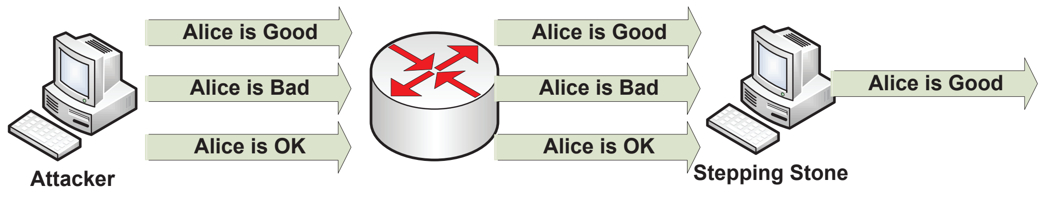
The topological analysis show all but one of the expected flows in a graphical format. The only flow not detected is an off-site host because the flow does not closely correlate to the flows of the local bots. This demonstrates that the scheme is effective at detecting local bots, but it cannot correlate the traffic to off-site hosts even if the C2 server was local.

2.6 Error Correction and Coding

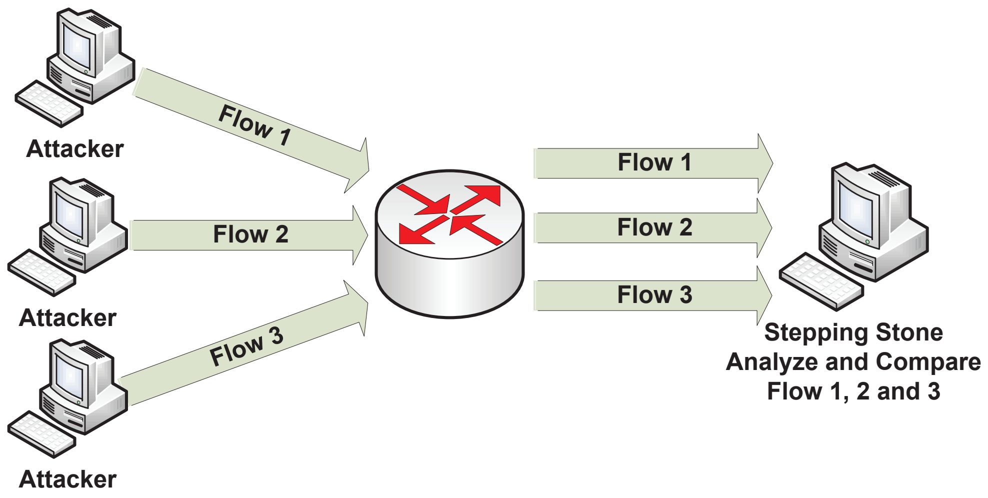
The watermarks must be present even when there is noise and distortions of the original data. Therefore, coding theory for error detection and correction offer an opportunity to increase the detection rate while maintaining the same encoding and decoding algorithms. As some of the SSD techniques use error correction and coding schemes, these applications directly apply to the future work done in this area. The RA code is an example that is fairly low complexity and easy to derive. Demonstrated in 1998, RA is a simple and low complexity code that provides good performance using Maximum-Likelihood (ML) decoding [4]. Although the complexity of the ML decoding is prohibitively large, it is also shown that the “turbo-like” decoding which approximates the ML decoding also performs well. In the case of the RA code, an information block of size n_A is first repeated q times. The $q \times n_A$ data is then scrambled by a permutation function and encoded by a rate 1 accumulator. The RA code rate is $1/q$ with a linear encoding time, making the technique fast.

2.7 Evading Detection

Before describing the various attempts to identify stepping stones, a discussion on the various techniques attackers employ to evade stepping stones is necessary. The different evasion techniques are important to understand as they significantly impact the development of the algorithms to identify the stepping stones. The properties of evasion are also critical to understand the effectiveness of the SSD techniques. Some of the more obvious techniques are encrypting the data as well as spoofing packets. While fairly trivial to implement, these do not offer much resilience against detection. More advanced techniques include timing perturbation as well as packet chaffing. In timing perturbation, the individual packet timing is adjusted so that the correlation between the incoming and outgoing stream is more difficult. Depending on the level of perturbation, this correlation is still possible using current SSDs. Packet chaffing is simple in design as shown in Figure 2.3(a), but not trivial to implement [5]. The changes present an evolution in the stepping stone from being a simple pass thru device to a more active filter. Indeed, the method of evading detection presents new challenges to the SSD problem.



(a) Packet Chaffing



(b) Multi-Flow Attack

Figure 2.3: Examples of Evading Detection

In addition to evading detection, the attacker may implement active techniques to deceive the detector. Assuming the attacker controls the stepping stone, there exists a possibility of modifying the network packets via a multi-flow attack as shown in Figure 2.3(b). For example, if the attacker controld the stepping stone and establishes multiple flows of traffic into the host, then analysis of these flows could be performed. If any deviation is exhibited from the original traffic patterns, then the attacker may assume the presence of a watermark and can lead to the identification and removal of the watermark.

2.8 Summary

The importance of detecting stepping stones, whether the attackers use botnets or simple SSH commands, is very difficult. However, detecting stepping stones in real time is invaluable to a network defender responsible for protecting a computer network. While the passive based approaches show promise with less mature attackers, the increased complexity of the attacker coupled with the computational resources needed make these methods inferior. Indeed, active based watermarking techniques have proven effective given the ever increasing network and attacker complexity. In the following chapters we present a novel active technique effective at identifying interactive sessions used in stepping stones.

3 Methodology

The methodology in this research consists of three main steps. First, timestamps are extracted and analyzed from previously recorded network conversations from the CAIDA 2009 [24] dataset. Next, after the watermarking technique is developed and refined, it is simulated using the previously captured CAIDA 2009 timestamps. Finally, the timestamps are used again to generate live watermarked traffic using stepping stones in the Amazon EC2 environment.

3.1 Problem Definition

3.1.1 Goals and Hypothesis. The main goal of this research is to effectively detect network stepping stones by applying a novel semi-blind active watermarking technique. A subset of the goal is to better characterize typical stepping stone network traffic. The system should be scalable with acceptable rates of detection. It also should not significantly change the characteristics of the original waveform.

Characterizing a “better” detection technique involves many factors including the FP and FN rates and the rate of detection relative to the number of network packets needed. The hypothesis is that by characterizing the selected Secure Shell (SSH) traces, a novel technique of watermarking the timestamps can be identified that demonstrates an acceptable level of accuracy and speed (e.g., number of packets or interval detection length) while retaining resistance against known attacks. Generally, the number of packets and time required to detect the watermark affect the accuracy. If the number of packets required for the higher detection rate is excessive, the technique may not be as effective. For example, if the system identifies 10% more stepping stones but it requires ten times the number of packets, it may not be suitable because many network conversations are short-lived.

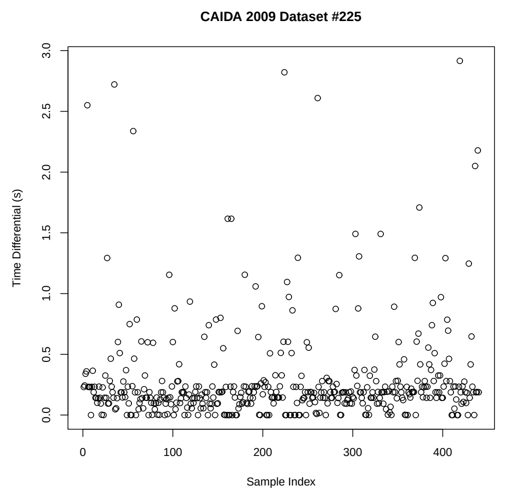
3.1.2 Approach. To improve stepping stone detection, this system illustrates a novel technique to watermark network packets. The main components of this research comprise characterization of the traffic, algorithm development, and system testing.

Using a selected set of typical stepping stone traffic from the historical SSH traces extracted from the CAIDA 2009 dataset [24], opportunities to exploit previously unknown aspects of the stepping stone traffic are used to improve the detection algorithm. The random samples of historical data are used for traffic generation during the experiments. The simulation model estimates the algorithm performance in a controlled, theoretical experiment.

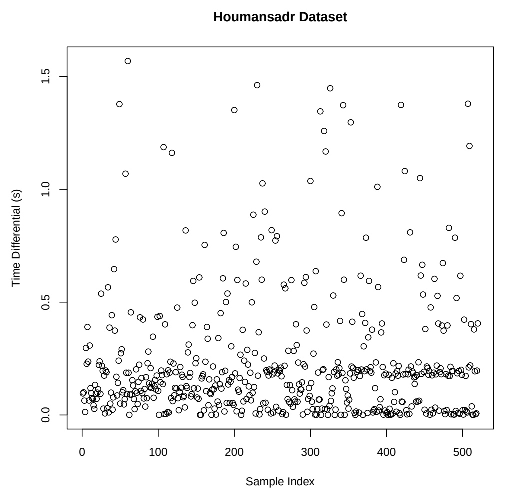
Once the model is analyzed and the algorithm is adjusted, a live experiment further verifies the performance. Often times, analytical models in networks have different variances when compared to live network traffic. For that reason, live performance analysis is critical to establish validity and demonstrate that the recommendations improve detection.

3.1.2.1 Analysis. The full CAIDA 2009 dataset includes over 2.4 Tebibyte (TiB) of network data without any payload information. Initially, tcpdump filters port 22 traffic relevant to SSH or Secure Copy (SCP) containing data. Data packets are the primary targets to modify using the active watermarking. The timing of management commands (i.e., TCP setup, acknowledgement and reset packets) is not conducive to modification because of required buffering which prevents atomic processes from completing. The simple tcpdump filters reduce the amount of information to be processed to approximately 5 gibibyte (GiB). Writing a program to parse the files and extract the timestamps requires multiple steps. Note that more specific details are provided in Appendix A. First, the program identifies the tuple of each packet containing the source Internet Protocol (IP), source port, destination IP and destination port. It then maintains a database of tuples identifying each individual network stream. Next, as the database of tuples fills to approximately 90%, it periodically filters out all streams containing fewer than a specified amount of timestamps. This removes the streams when a connection is attempted, but not actually made. In this research, the maximum amount specified for removing network conversations is fourteen packets in length. This prevents stale conversations from filling the database too quickly while retaining the longer lived conversations ideal for testing the watermark technique. The program maintains a database of approximately 50,000 entries and purges stale values when the capacity reaches approximately 90% capacity. The program records the timestamp of each packet when the destination port is 22, indicating SSH or SCP traffic. The final filter removes all traces containing an average packet size greater than one kibibyte (KiB). Typical packet sizes of SCP traffic average larger sizes, so the filter removes traffic that indicates SCP rather than SSH. The final file size for the extracted timestamps for the CAIDA 2009 dataset is approximately 140 mebibyte (MiB).

3.1.2.2 Comparison to Previous Data. Sixty traces from the Houmansadr data [8] (also extracted from the CAIDA 2009 January dataset) are compared to the data after parsing the files. Though the actual tuple that corresponds with the previous research is not provided, Houmansadr indicates that the traces are truncated so that they are all approximately 120 seconds in length. Both the extracted and the Houmansadr traces reveal that the typical SSH traffic inter-packet delays are generally bimodal and contains peaks at approximately 0.05 and 0.2 seconds. The plots generated previously show some indications of the bimodal inter-packet delays, but they also show that there are many plots in which there is a high rate of traffic without much deviation. Although Houmansadr references that he is able to extract over 300 useful SSH traces from the January CAIDA 2009 dataset, the filters described in Section 3.1.2.1 reveal only 99 useful traces for this experiment. Figure 3.1 illustrates the comparison of the inter-packet delays of the extracted data and a similar dataset provided by Houmansadr. Figure 3.2 shows the density plot of the inter-packet delays of the same samples. Note that Appendix B illustrates the unused traces that exhibit characteristics not consistent with human-generated traffic. These rejected traces show constant packet rates as well as extremely high rates of speed.



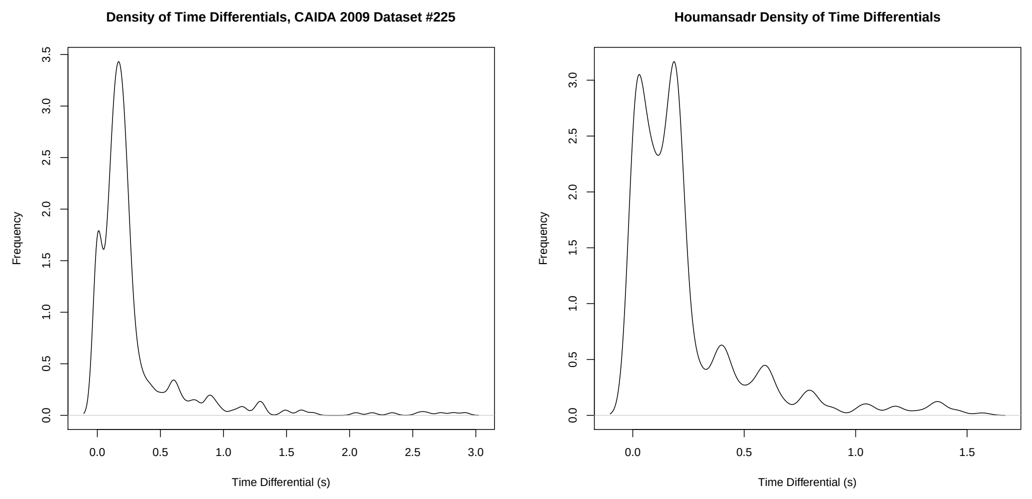
(a) Extracted Dataset Sample 225



(b) Houmansadr Example Number 4

Figure 3.1: Example Plots of Extracted Timestamps and Houmansadr Data

3.1.2.3 Inter-packet Delay of SSH Streams. The density plots of the inter-packet delays reveal that most of the traces appear to be bimodal. The bimodal distribution is not consistent with previous research that indicates the inter-packet delay is distributed according to a Poisson distribution; however, the distribution is consistent with recent research regarding the inter-packet delays. The bimodal characteristics of the differential time values are shown in Figure 3.3. A bimodal distribution does not lead to an obvious method to watermark as it is difficult to retain the underlying statistics, making the watermark invisible. Analyzing the differences between the inter-packet delays shows that it follows closely to a normal distribution. The analysis of the difference between the packet delays closely resembles a second derivative and progresses smoothly to an



(a) Density of Inter-packet Delays, Sample 225 (b) Density of Houmansadr Data, Sample 4

Figure 3.2: Example Density Plots of Inter-packet Delays for Extracted Data and Houmansadr Data

analysis using DWT techniques. It is important to note that this analysis is different from a lag -2 plot, which shows the difference between every other packet. Instead, this analysis illustrates the differential analysis of the timestamp differentials.

3.1.2.4 Discrete Wavelet Transform. As indicated in the background, previous multimedia watermarking techniques use the DWT. The fundamental waveform for the DWT is the Haar transform and is applied to the SSH time differentials. The values indicate that the first three vectors are approximately normally distributed around zero for all of the SSH traces. This leads the algorithm development to use first detail vector d^1 of the DWT to encode the watermark values using the sign of the vector. Pilot experiments result in the final algorithm shown in Figure 3.4 and the detector in Figure 3.5. Note that the synchronization frame and parity check are features that correct problems later observed after adding simulated jitter.

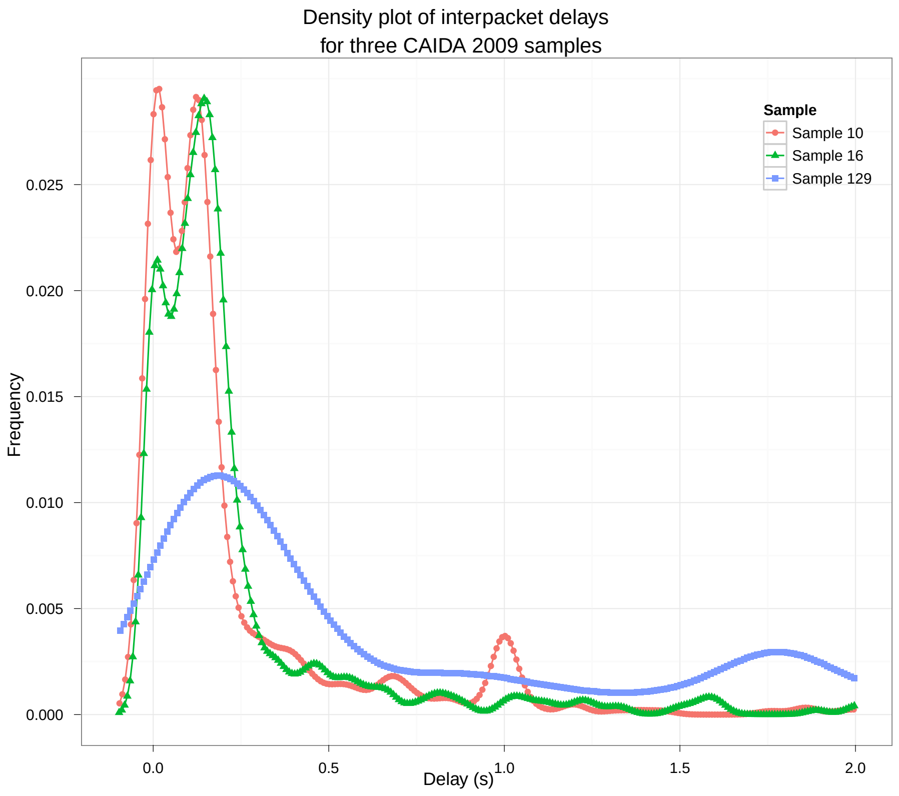


Figure 3.3: Distribution of Differential Time Intervals for SSH Streams. Three values chosen from the 2009 January CAIDA dataset to show bimodal characteristics.

3.2 System Boundaries

The System Under Test (SUT) consists of many components. The SSD includes the watermarker and the detector as well as the network equipment and hosts. The dotted lines in Figure 3.6 reflect the logical boundary of the system. While the components may be physically or geographically separated, the mechanism performing the active watermarking and the component responsible for detecting the watermarks define the logical boundaries.

In the simplest design, the SUT is self-contained in the protected network. In more complex designs and with appropriate permissions, the watermarker and detector do not

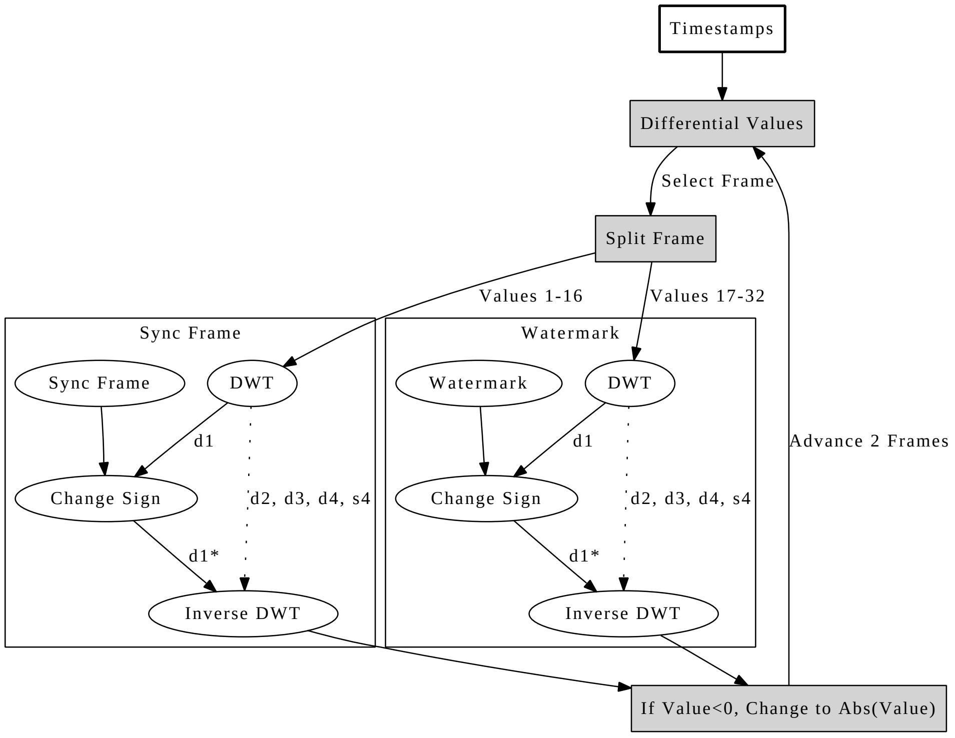


Figure 3.4: Algorithm of the Watermarker

need to be on the same physical or logical network. With the proper parameter selection, the detection mechanism is robust against variances caused by the network.

The main Component Under Test (CUT) is the watermark detection algorithm. While both the watermarker and the detector perform two different operations, the actual CUT is the detection algorithm. This research builds on previous designs and tests conducted to achieve a high level of detection with the fewest number of packets or time. Immediate and accurate identification of the stepping stones are the most important characteristics of a system that is operationally useful to network defense. This research introduces a novel watermarking technique and evaluates the effectiveness.

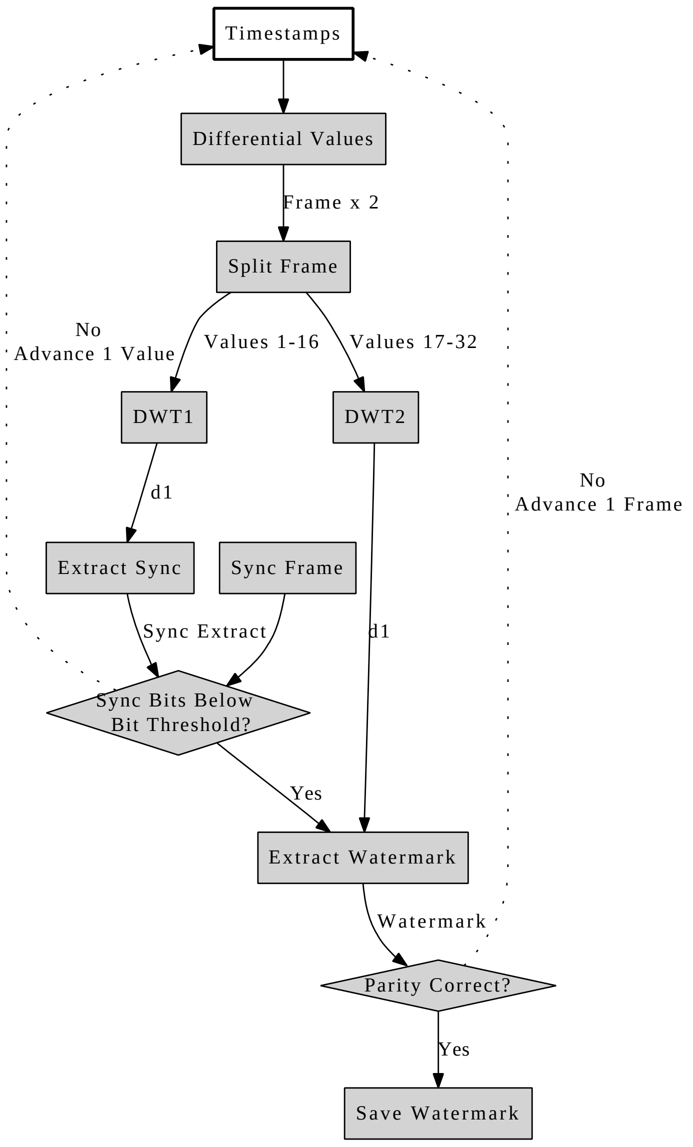


Figure 3.5: Algorithm of the Detector

3.3 System Services

The SSD provides active network watermark detection in the presence of robust countermeasures without significantly affecting the network characteristics. The success of the system primarily resides in accurate stepping stone detection. Given a null hypothesis that there is no watermark present at the detector, the outcomes of the detector fall into one of four categories:

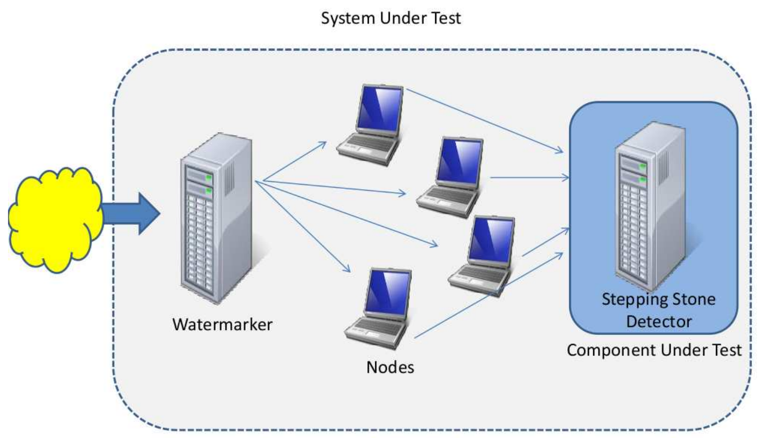


Figure 3.6: The System Under Test and Component Under Test

- True Positive (TP): The detector correctly identifies a watermarked stream as watermarked.
- False Positive (FP): The detector falsely identifies a non-watermarked stream as watermarked.
- True Negative (TN): The detector correctly identifies a non-watermarked stream as non-watermarked.
- False Negative (FN): The detector falsely identifies a watermarked stream as non-watermarked.

The watermark presence is determined at the detector by the watermark statistic τ. This is calculated as a ratio of successfully detected synchronization frames divided by the number of total frames. Comparing this statistic to the detection threshold γ determines whether a watermark is present. Adjusting the threshold to a higher value (i.e., closer to 1) results in less FP values, but increases the number of FNs. Lowering the threshold (i.e., closer to 0) generates more positively identified watermark streams, increasing the FPs while lowering the FNs.

Another system service is the modified network stream. The modifications of the inter-packet delays may indicate to an attacker that the stream is altered. Previous research indicates that this watermarking technique is resilient to cryptanalysis and the distribution of the watermark data is nearly identical to the original.

Significant system failures represent a critical reduction in the effectiveness of the system in terms of the overall goal. A false negative detection represents an active stepping stone that is not detected and therefore can pass freely in the network. While the attacker may need additional measures to remove a watermark, the active removal of a watermark by the attacker is also a critical system failure.

The sub-optimal performance measures demonstrate system failures in which the system may not perform ideally, but the goal of detecting the stepping stones when present is still maintained. False positive detection may require human intervention to filter the false detections and can degrade overall performance if it is too severe. The detectable watermark and network interference both provide the stepping stones with additional information regarding detection tactics, but these do not represent critical failures of the system to detect the stepping stones.

3.4 Workload

The stepping stone traffic has factors that are considered system workloads. The packet size, throughput and inter-packet delay are important factors that significantly alter the performance of the algorithm. The primary factors affecting the algorithm are the inter-packet delay distribution and packet inter-arrival rate. Because the detector is fundamentally based on the positioning of packets, carefully chosen parameters are often only effective against traffic that follows a certain workload. If the algorithm changes in any way, other factors may have greater influence. The primary factors in this system are the stepping stone traffic throughput and inter-packet delays.

In this experiment, the workload factors are chosen by randomly selecting previously recorded SSH traffic flows from the CAIDA dataset as system inputs. The packet timings are watermarked and input to the detector. To analyze the FP detection, random unmarked packet streams bypass the watermarker and are input directly into the detector. The simulation and live experiments use a random selection from the CAIDA 2009 dataset.

3.5 Performance Metrics

In the SSD, the primary goal is to detect network stepping stones. Indeed, accurate detection without significantly altering statistical distributions are the primary factors in optimizing the algorithm. The primary metrics are significant to identifying the presence of a watermark and correctly extracting each encoded bit. Secondary metrics not essential to this system are addressed, but are not the main focus of the research. The primary performance metrics to evaluate the SSD are:

- False Negative Detection Rate
- False Positive Detection Rate
- Correctly Extracted Watermark Bits

The FN rates are essential metrics in evaluating the accuracy of the detection system. From an operator workload perspective, the FP rate is equally important to identify the detections that are not true stepping stones. The bits detected per watermarked stream are important to identify the accuracy of the system as well as the detection speed. Although the threshold determines whether the entire stream is detected as watermarked, the number of bits correctly detected per watermarked stream is also important in correlating the stepping stones.

3.6 System Parameters

All of the parameters that affect the performance of the system can be characterized as workload parameters and system parameters. These are shown in Table 3.1. Figure 3.7 also shows the SUT with the parameters, workloads and outputs. The system is most sensitive to the watermark interval length directly correlating with the number of timestamps in each stream. The frame size determines the amount of timestamps necessary for the watermark and must be a multiple of the number of encoded bits. In this experiment, the system encodes seven bits with one parity bit, so the frame size must be at least 16 (i.e., $(7 + 1) \times 2$). A frame size of 32 is also tested in the simulation. The bit detection level determines the number of correct bits required to identify each synchronization frame. Typically this value is either 0 or 1 in the pilot studies, but increasing the value allows for the detection in the presence of greater noise. The minimum value of 0 decreases the likelihood of a false positive, but it may pass over actual synchronization frames if there is a large amount of network noise. The frame size is expected to affect the system, but not as much as the bit detection level. The threshold is selected to minimize the FP and FN rates and the rest of the parameters have a low sensitivity to the system.

Table 3.1: System and Workload Parameters

System Parameters	Workload Parameters
Watermark Interval Length Frame Size (s) Bit Detection Level (T) Watermark Threshold (γ)	Previous SSH trace to be watermarked Previous SSH trace not to be watermarked

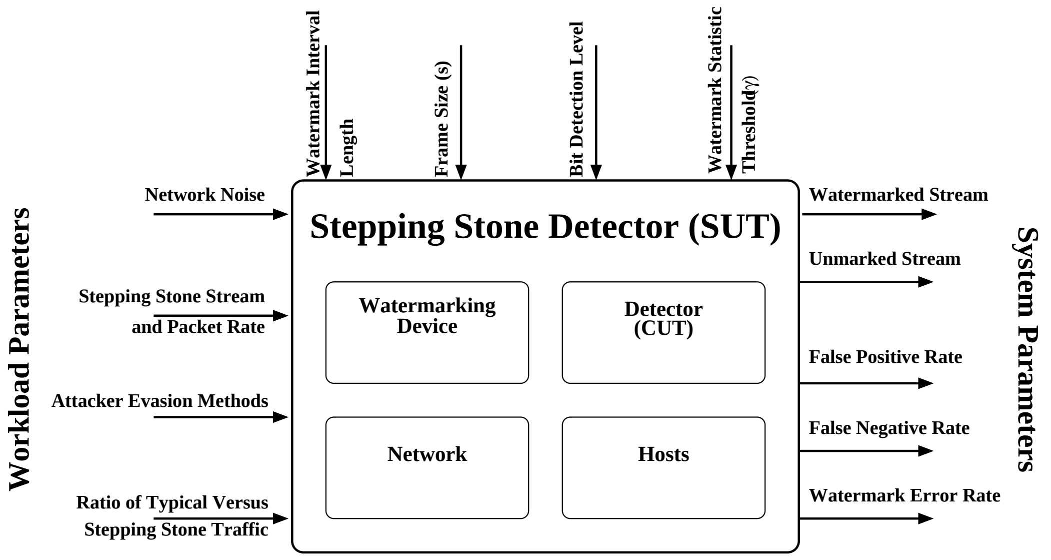


Figure 3.7: System Inputs, Parameters and Outputs.

3.7 Factors

In concert with the primary motivation for this research, the system parameters involving the watermarking and detection algorithm are the main factors to be studied. The algorithm factors include the selection of the bit detection level, frame size, and watermark threshold (γ). In addition, factors during the simulation include the simulated noise (jitter) and the network path for the live experiment.

These factors shown in Table 3.2 are the subset of the system parameters evaluated in the simulated experiment based on expected performance measures. The simulation results confirm whether the levels should be maintained in the experimental design or if they should be modified. A preliminary threshold for the live experiment is also determined during the simulation. The simulation compares the watermark statistic to the expected theoretical value in order to further validate the expected results and ensure the model is accurate. The simulated jitter levels are selected to be similar to previous research by Houmansadr. Other research shows that worst case jitter values range around 5 ms [14], so these chosen values represent values beyond previous research extremes.

A comparative analysis used for validation is performed using both the extracted SSH traces as well as 60 traces provided by Amir Houmansadr as part of the SWIRL research. The extracted dataset provides the fundamental values; the Houmansadr traces are used to validate this research against previous watermark research.

Table 3.2: Factors for Simulated Experiment

Factors for Experiment	Levels
Simulated Jitter	6.2-12 ms Standard Deviation
Detection Threshold	0, 1 bits
SSH Trace Data	Watermarked Traces, Unmarked Traces

Table 3.3: Factors for Live Experiment

Factors for Experiment	Levels
Network Path	Tokyo, Ireland, California, Virginia
SSH Trace Data	Watermarked Traces, Unmarked Traces

3.8 Evaluation Technique

The two evaluation phases of the SSD are simulation and a live experiment. The simulation watermarks the timestamps of a random extracted CAIDA SSH trace. The client generates packets at intervals based on these timestamps. The packets timings are then modified using Gaussian noise and input to the detector. The watermarker and detector are simulated in a statistical package called R. The results determine the interaction between factors and also estimate the amount of replications to run in the live experiment.

The live experiment uses the same random sampling of traces, but the output of the watermarker and detector are performed using hardware implementations. Because each random sample is approximately two minutes in length, the number of replications in this phase is less than the analytical phase. Each of the computers in the live experiment are virtualized t1.micro Amazon EC2 instances. The configurations of the packet generator, repeaters, and the receiver are listed in table 3.4. The packet generator and receiver code is written in Python using the Twisted module version 11.0.0 and Scapy module version 2.1.0.

A graphical depiction of the live server locations is shown in Figure 3.8. The black lines represent packets generated by the client and the red lines show the direction of the stepping stone traffic. The California server generates all of the live traffic and sends it to one of the repeaters or the final destination in Virginia. The repeaters forward the traffic on to the next server and ultimately all end in Virginia. The network path for the repeaters is Tokyo to Ireland to Oregon to Virginia. Four separate processes on the client generate independent data streams simultaneously during the experimental phase. The combination of marked and unmarked traffic more accurately simulates noisy network environments.

Table 3.4: Live Experiment Hardware in Amazon Elastic Compute Cloud

System	Location	Zone	AMI	Linux Distribution
Client	California	us-west-1b	ami-1bd68a53	Red Hat Linux (64-bit)
Repeater 1	Tokyo	ap-northeast-1a	ami-0644f007	Red Hat Linux (32-bit)
Repeater 2	Ireland	eu-west-1c	ami-953b06e1	Red Hat Linux (64-bit)
Repeater 3	Oregon	us-west-2a	ami-38fe7308	Red Hat Linux (32-bit)
Server	Virginia	us-east-1a	ami-60ee1109	Ubuntu Linux (32-bit)

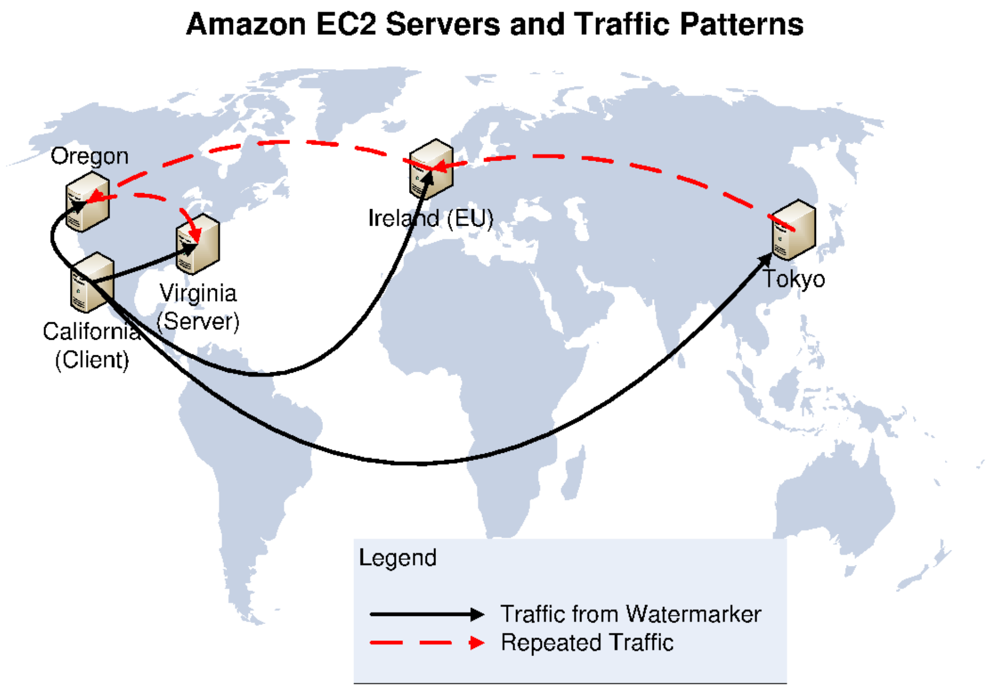


Figure 3.8: Amazon Elastic Compute Cloud Locations

The validation strategy includes a cross-validation of the simulation and live experiments and a comparison against previous research. The live experiment results should match closely those predicted from the simulated model. Comparisons using the previous Houmansadr data to the SWIRL design also help validate the research against another watermarking technique.

3.9 Experimental Design

The experiment requires a full-factorial of the parameters to quantify the interaction as well as a substantial amount of replications to allow for the jitter levels. Based on previous research, approximately 1,000 replications create a sufficient statistical basis for analysis and the pilot studies determine the statistical significance of these results.

Therefore, approximately 24,000 total iterations are needed to evaluate the system performance and validate the system against previous research.

The live experiment requires a full factorial of the servers (i.e., Tokyo, Ireland, Oregon and Virginia) and the marked and unmarked data. The extracted SSH traces are selected at random for each sample. Based on the pilot studies, approximately 100 samples of 120 seconds in length are required to estimate the FP and FN values for each server. The amount of repetitions is based on the variance of the data. Assuming four independent sessions consisting of three watermarked and one unmarked session, the total number of trials per repetition is $4 \times 4 \times 100 = 1,600$. A total of five repetitions results in 8,000 trials in the live experiment. Assuming each trial is approximately 120 seconds and four processes run simultaneously, the live experiment takes about 67 hours.

The model confidence level is 95% to establish factors before a hardware implementation is considered. Before progressing to the live experiment, the simulation should show that the watermarked and unmarked statistics are different at a 95% confidence level or better. Because of the effort required to build and run the live network model, a 95% confidence level ensures that the model can determine within the specified statistical accuracy, that the performance is in fact superior. The variance in the live experiment may be greater than expected, but the results should still show a difference between the marked and unmarked streams at a 95% confidence level or better.

3.10 Methodology Summary

This research analyzes previous network traffic and introduces a novel watermarking technique. The traffic characterization uses statistics to analyze previous captured network traces from the CAIDA dataset to characterize traffic typical for a stepping stone. The research analyzes inter-packet delays for the different streams and determines the characteristics of the traffic to better optimize the detection parameters.

The extracted timestamps are used as data inputs to test the algorithm using a computer simulation in R. The results of the simulation establish the baseline for validating the algorithm and the live experiment. Factors such as parity checking and framing are evaluated in the algorithm to determine interactions and achieve a more optimal performance in the live experiment. The simulation also sets an initial baseline for the threshold value used to determine if a watermark is present.

Finally, the traffic generated by a live computer is passed through various stepping stones across the Amazon EC2 to determine true rates in live environments. This substantiates the analysis in the simulated model. This technique is expected to detect the presence of the watermark with predictable error rates using similar stream sizes (i.e., length of time) to the SWIRL design . The values extracted by the detector are also expected to match the watermark sent by the client.

4 Analysis and Results

In this chapter, the traffic analysis of the CAIDA 2009 dataset is first presented. The features are used to develop the algorithm of the watermarker and the detector. Next, the simulation results demonstrate a statistical difference between the watermarked data and the unmarked data. Based on results, a threshold is determined to detect the presence of the watermark. Finally, the analysis of the live simulation demonstrates that this technique is a viable watermarking algorithm for certain types of interactive traffic.

4.1 Traffic Analysis

Analysis of timestamps reveal important characteristics. First, Figure 4.1 illustrates the first level Haar wavelet decomposition vectors d^1, d^2, and d^3. Each of the vectors are approximately normally distributed around zero. Second, the density plot of the timestamps appear bimodal. The Haar observation is useful in applying the watermark using the DWT because the watermark must retain the bimodal characteristic for invisibility.

The Haar wavelet decomposition is performed using the *waveslim* library in R. The first vector represents a scaled value for the difference of the pairs of time differentials. For example, using the time differentials $n_0, n_1, n_2, \cdots, n_j$ the values of the first vector d^1 is as shown in Equation 4.1.

$$d_n^1 = \frac{-1}{\sqrt{2}}(n_i - n_{i+1}), n = 1, 3, \cdots, n-1 \tag{4.1}$$

The plot of the first three vectors for the four level Haar decomposition using 32,768 unique timestamp differentials is shown in Figure 4.1. As shown in the figure, the values for the first vector are approximately normally distributed around zero. This discovery of the vector distribution originally led to the realization of the watermark application using the sign of the vectors. Performing an inverse DWT on the manipulated d^1 vector while retaining the remaining vector has an effect similar to interchanging the packets in the time domain. This is graphically depicted in Figure 4.2.

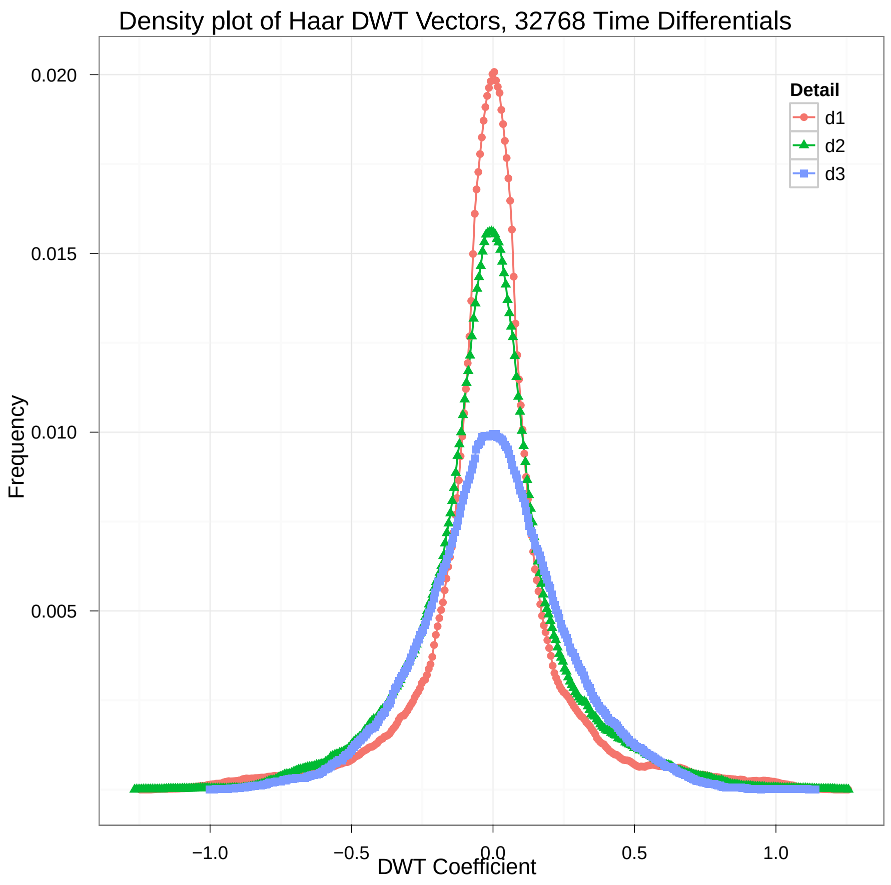


Figure 4.1: Distribution of the first three detail vectors for the four-level Haar Discrete Wavelet Transform using time differentials as inputs.

4.2 Algorithm Development

4.2.1 Frame Length. The discrete wavelet transform vector d^1 contains exactly half the number of values as the input vector. Thus, at most the watermark algorithm encodes half of the number of bits selected for the frame size parameter. To use an 8-bit

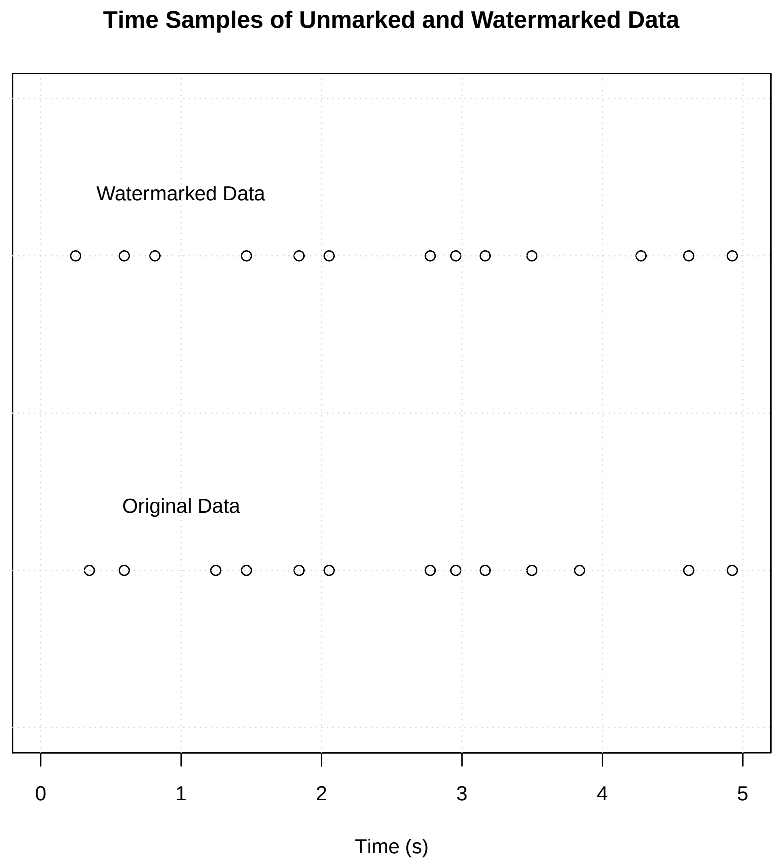


Figure 4.2: A graphical illustration of the watermark applied to a sequence of time packets. A sign change in the d^1 vector appears like two timestamps changed places in time.

watermark, the frame needs to be at least 2^n where n is an integer greater than three. The transform relies on an accurate synchronization of the frame because the transform generates the vector values based on the differential time pairs. If even one timestamp is lost, all differential time pairs are modified from that point until the end of the frame. This requires that the frame is tightly synchronized and that any losses or errors are identified and corrected quickly. The requirement leads to the generation of the synchronization frame for each watermark. Although this data is redundant, it allows for identification and in some cases correction of errors in the presence of network noise and packet loss.

4.2.2 Synchronization Frame. The algorithm is also dependent on the choice of the synchronization frame. Because the watermarker and the detector need to use the same value in the synchronization frame, the system is labeled as semi-blind. The discoveries during the experimental runs show the characteristics of a synchronization frame that is more robust than others. First, if the synchronization frame is 8 bits, then the two four bit words should be different. The reason for incorrectly identifying the synchronization frame in these cases is due to the random watermark. If the synchronization frame contains bits ABCDABCD, the detector is more likely to incorrectly identify the synchronization frame if the watermark contains the same word in the first or second positions. In other words, if one word of the randomly chosen watermark matches the synchronization frame (i.e., ABCDEFGH or EFGHABCD), the detector incorrectly identifies more synchronization frames than is predicted at random. In addition, the synchronization frame should contain an equal number of 1 and 0 bits. This retains the original distribution centered around zero of the Haar vector d^1. The chosen synchronization frame for the experiment is 1100 1010.

4.2.3 Frame Size. The DWT requirement of 2^n values per frame requires the encoding of 8 bits to have a frame size of $8 \times n$ where n is an integer greater than 1. The larger frame size encodes the synchronization or watermark as rows of 8 inside the larger frame vector. Increasing the frame from 16 to 32 values does not increase the detector performance in terms of speed or accuracy. In fact, in some cases the 32 value frame makes it less resilient to errors that occur near the beginning of the frame. Assuming that the errors occur at random, the 32 value frame does not pose an advantage over the 16 value frame. Pilot studies indicate that the 32 value frame does not offer a significant performance improvement over the 16 value frame. In a physical implementation, it is also more difficult to modify a larger frame size because of buffering. For these reasons, the live experiment only uses a frame size of 16 to increase the resiliency to lost packets.

4.2.4 Threshold. To detect the presence of the watermark using a semi-blind detector, a calculation needs to be made to determine if a watermark is present. The technique requires the synchronization frame to be sent prior to each watermark. The presence of the synchronization frame preceding each watermark combined with the distribution of the Haar d^1 vector allows the threshold to be a direct calculation of the number of synchronization frames detected. Assuming that a non-watermarked stream d^1 vector follows the distribution given in Figure 4.1, the likelihood that the detector will find a synchronization frame at random is evaluated in Equation 4.2, where T is the bit detection level for an 8 bit synchronization value. The bit threshold is best described as the number of accepted bit errors for the detector to correctly identify the synchronization frame. Note that setting $T = 0$ means a synchronization frame is recorded only when all eight bits are correctly detected as in,

$$P(T) = \sum_{0}^{T} (\frac{8}{8-T})(0.5)^{8-T}(0.5)^{T}. \tag{4.2}$$

For $T = 0$, the chance of a synchronization frame exactly occurring is 0.0039. When $T = 1$ for use in noisy environments, the probability increases from the zero bit value by a factor of 10 to 0.03516. These analytical thresholds are confirmed using 60 unique time differential arrays of over 3,000 data points. The number of times the previous synchronization frame exactly occurs ($T = 0$) is 0.0038832 (i.e., 1563 occurrences in 402500 unique 16 value frames). Setting $T = 1$ also yields similar results with a probability of 0.035098 (i.e., 14127 correct in 402,500 unique 16 value frames).

4.3 Watermark Application and Invisibility

A density plot in Figure 4.3 shows an unmarked sample, the same sample after watermarking and the watermarked sample with noise added. In this case, the additive noise is the maximum 12 ms and demonstrates a significant difference between the original and watermarked sample. It also shows how closely the watermarked sample resembles the unmarked sample. This is due to the algorithm shifting the inter-packet delays and its ability to preserve nearly all of the inter-arrival densities. The only time differentials that change from the original are those in which the negative watermarker output is changed to the absolute value. This occurs in less than 0.1% of the total number of packets. The detector is still able to correctly extract the data encoded in the watermarked stream with additive noise as shown in Figure 4.3.

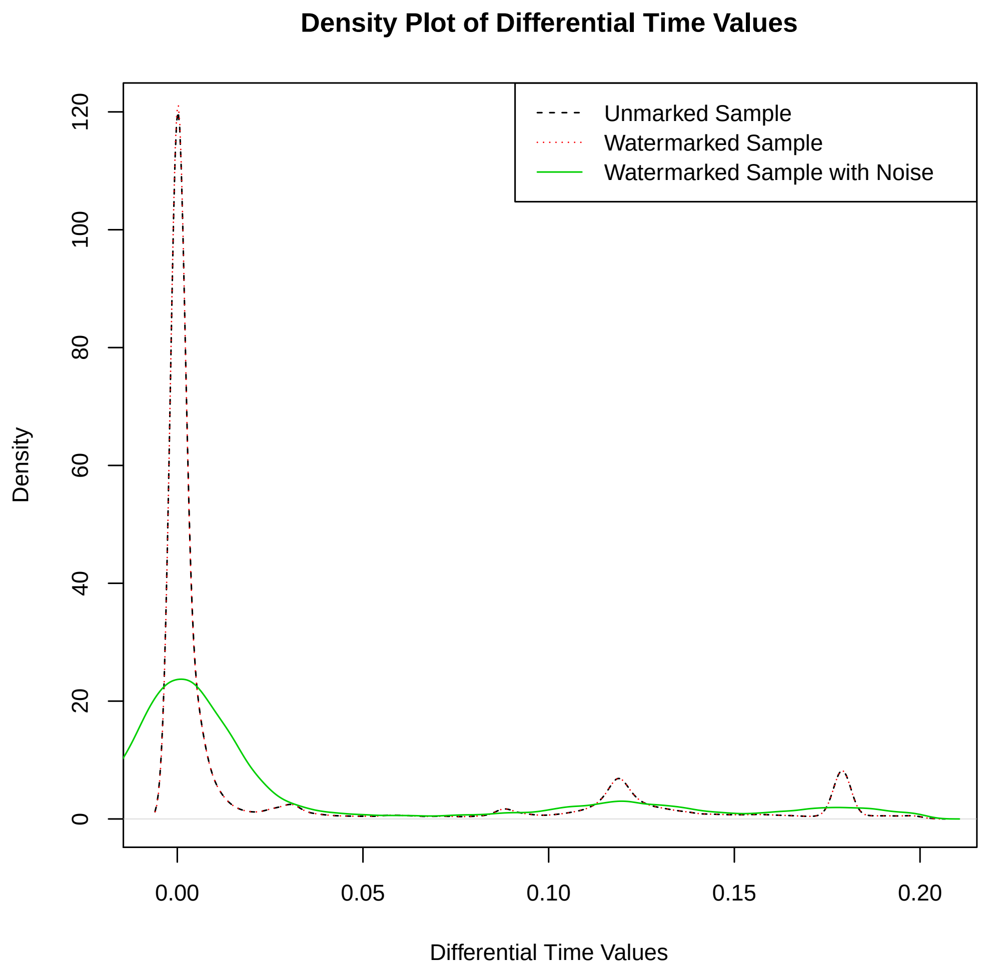


Figure 4.3: Density plot illustrating the unmarked, watermarked and watermarked sample with simulated noise.

4.4 Simulation

4.4.1 Pilot Studies. The pilot studies using the R simulation show that the frame size of 16 is more optimal for the detector. By reducing the frame size to the smallest value allowable while encoding 8 bits, the number of synchronization frames increases and enables the detector to more accurately decode the watermarks.

4.4.2 Threshold determination. The ability to determine if a watermark is present depends on a suitable choice for the threshold γ. In this case, γ is not analytically determined, but experimentally derived by the distribution of the watermarked and unmarked data in the presence of noise. The output of the simulation determines that the 95% quantile is approximately the point at which γ is equal for the unmarked and the marked statistics. For the purposes of this experiment, the 95% quantile for the unmarked data statistic is 0.01042 and the 5% quantile for the watermarked data statistic is 0.01045. The actual threshold lies between these two points, and for the purposes of this experiment, the mean of the two values is used in which $\gamma = 0.01043$. Note, future research may be able to analytically derive this value based on the rate of traffic as well as the statistical properties of the DWT vector d^1.

The watermark estimated probability densities are shown in Figure 4.4. The threshold value of $\gamma = 0.01043$ is also shown. Note that these densities are estimated using a kernel smoothing function and scaled to appear on the same graph. This causes the threshold to appear at a higher value than when the two densities cross on the graph. Also, the watermark statistic must be positive although this graph depicts otherwise because the density is only estimated.

4.4.3 Performance of the system. The simulation randomly selects a CAIDA 2009 extracted dataset and randomly applies network jitter before applying the detection algorithm. The system then tests for the presence of a watermark in these streams and

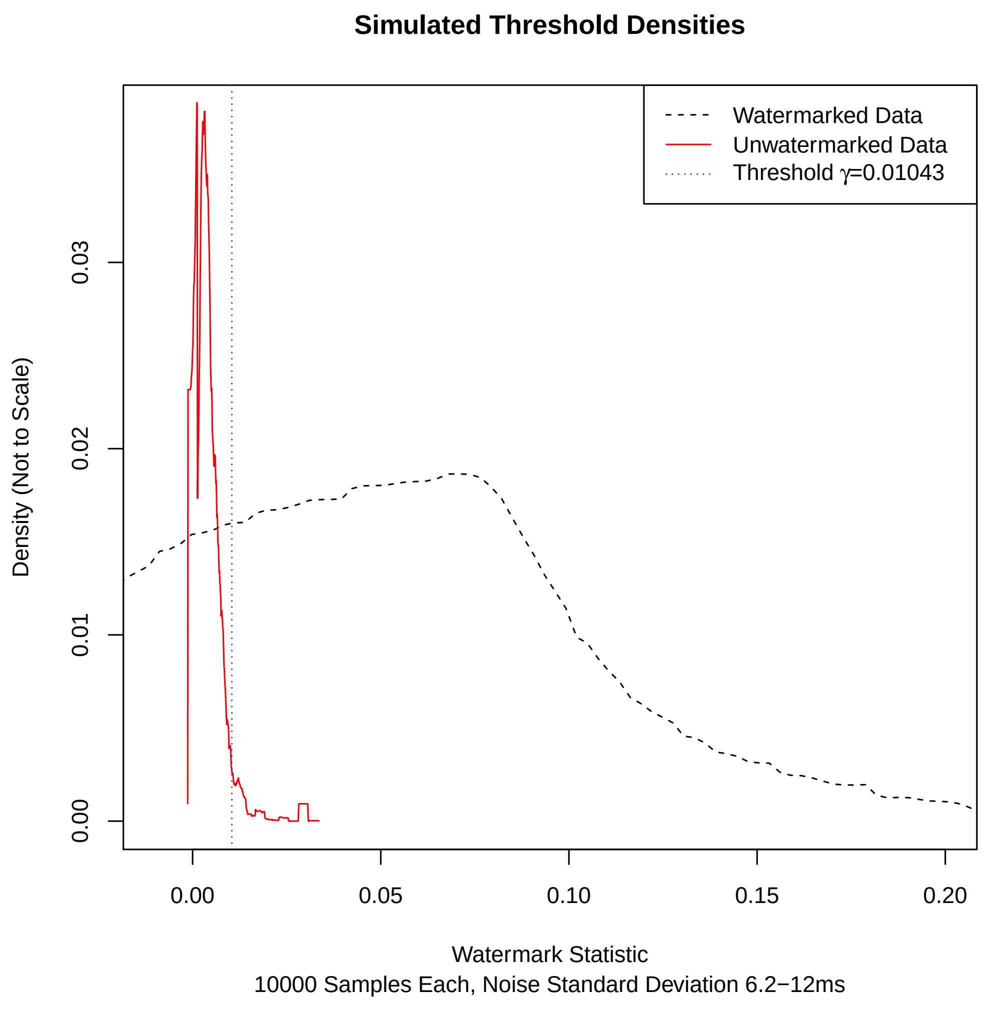


Figure 4.4: Density plot illustrating the estimated probability distributions of the watermark statistic. The threshold of $\gamma = 0.01043$ is also shown.

unmarked streams. The results show that using the threshold $\gamma = 0.01043$ at a 95% confidence level, the FP rate is 4.6-5.1%, FN rate is 3.96-4.44%, and the error rate in which the extracted watermark is incorrect is 7.86-8.30%.

4.4.4 Comparative Analysis. The simulation also validates the data rates compared to the previous research. Using the 60 Houmansadr samples, the error rates are drastically lower than using the 99 extracted CAIDA streams. In the simulation, the 60 samples produce similar results using comparable amounts of network jitter. In the case of the SWIRL test, live jitter values results with a standard deviation of 6.2-12 ms are replicated in the simulation. For this research, instead of using previously recorded values, the jitter is simulated as a Gaussian random variable with a mean of zero and standard deviation of 6.2-12 ms. In the case of the DWT design, the error rate in which the extracted watermark is incorrect is 1.57-2.97 per 10,000 at a 95% confidence. The FN rate using the Houmansadr samples is 7.8-8.4 errors per 1,000 samples, which is greater than the SWIRL rate of 10^{-6}. The FP rate of the simulation is closer to the SWIRL system with a rate of 0-7 errors in 100,000 samples. The results of the simulation using Houmansadr's values demonstrate a response in concordance with previous research.

Using alternate data by synthesizing streams based on the tcplib distribution also validates the performance of the algorithm. In this scenario, the simulation randomly generates 120 second timestreams using the tcplib distribution and determines the performance. Randomly selecting the 6.2-12 ms jitter, the watermark extraction error rate for the tcplib distribution is 6.97-7.28% at a 95% confidence level. This matches very closely with the simulated values from the CAIDA 2009 extracted dataset thus further validating both the simulation results and the accuracy of the CAIDA 2009 extracted timestamps.

4.5 Live Experiment Results

The results show that the watermarked data is more difficult to discern from unmarked data as the traffic traverses more paths. This follows intuitive knowledge that as the path gets increasingly noisy, the calculated statistic draws closer to the threshold γ and becomes more like the unmarked data. Sending the data from California through Tokyo, Ireland and Oregon before the server receives it in Virginia adds additional network noise such as jitter, thus generating more errors.

4.5.1 Watermark Statistic Analysis. To determine whether a watermark is present, the detector calculates the value γ by analyzing the number of synchronization frames present and dividing it by the total number of frames. In the case of a perfect network path with no noise, the number of synchronization frames will be exactly half of the number of total frames in the stream. For unmarked streams, the threshold will be closer to that determined using Equation 4.2 depending on the bit threshold. For a bit threshold of zero, the synchronization frame must exactly match and the unmarked statistic should be approximately 0.003.

Figure 4.5 shows the density of the watermark statistics over the course of the experiment. The unmarked streams to each target are grouped together and closely match the expected results in which the synchronization frame is detected according to random chance. The statistic mean for the watermarked streams increases as the target approaches the final destination. The Tokyo server traverses the most connections and, as expected, is less discernible from the unmarked streams. Likewise, sending traffic directly from the California server to the endpoint (i.e., Watermark Virginia in Figure 4.5) generates statistics with a higher mean than the other targets.

4.5.2 False Positives and False Negatives. The FP values occur when the detector identifies the unmarked stream as containing a watermark. The results confirm the expectation that the network path should not affect the error rates. Figure 4.6 shows the density of the unmarked streams for each of the targets. The Wilcoxon test also confirms that there is no statistical difference between the watermarked and unmarked streams.

The FN values behave much different than the FP. The FN values are dependent on the network characteristics, while the results show that the FP values are not dependent on the network path. The network noise alters the time delays in a manner that lowers the statistic mean such that the detector fails to identify the presence of the watermark.

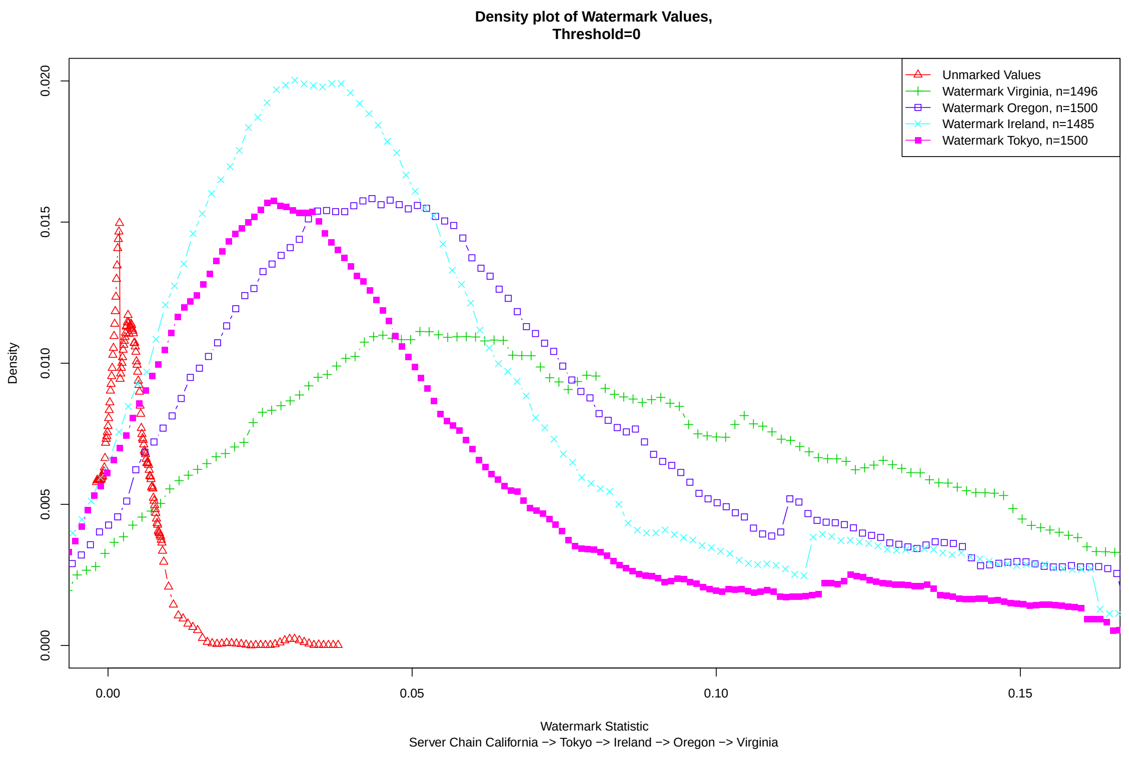


Figure 4.5: Density plot of Watermark Statistics over all servers. Unmarked data from all servers is grouped into unmarked values.

Excessive jitter, noise and delay are factors contributing to the FP rate. In the experiment, the Tokyo server generates the greatest number of false negatives while the closest server in Virginia generates very few. This is expected because the Tokyo server traverses the largest path from California through Tokyo, Ireland and Oregon before reaching the target in Virginia.

4.5.3 Dataset Anomaly. In the case of the graphs depicted in this research, it is important to identify and provide justification for an identified data anomaly. Figure 4.5 shows a small peak in the unmarked data at approximately 0.03. Figure 4.6 also shows that this anomaly is present in datasets sent to all servers. Upon further investigation, these data points originate from index number 99 of the extracted data. This dataset contains

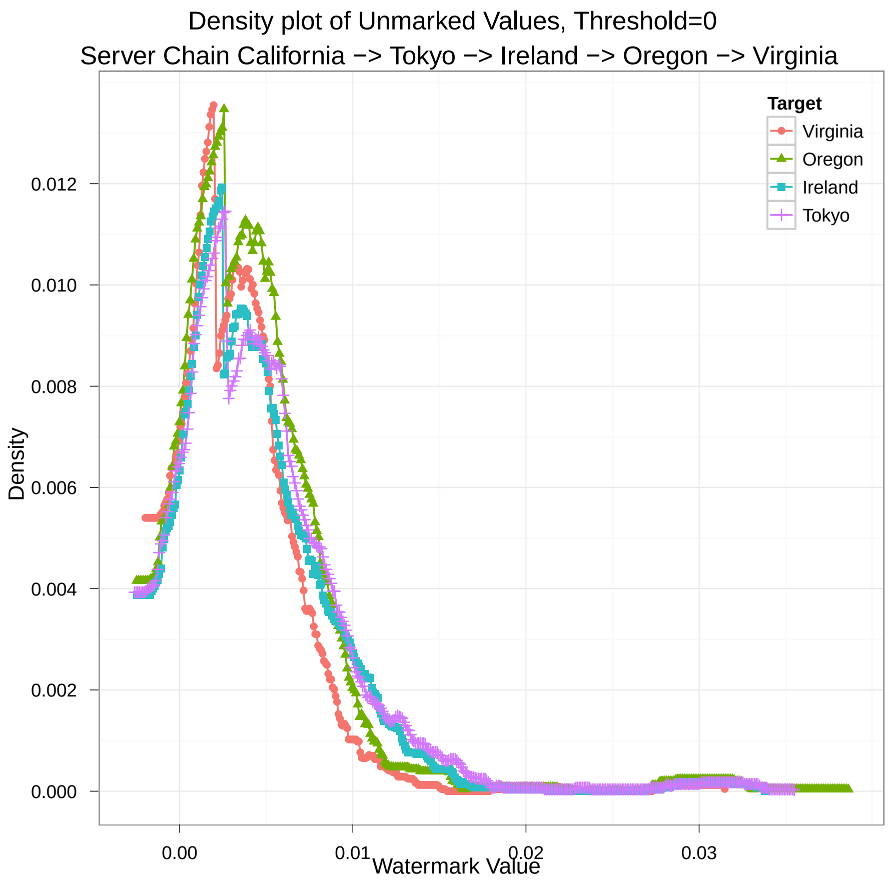


Figure 4.6: Density of Unmarked Live Experiments.

only 121 time differentials, resulting in a maximum of three synchronization and data frames to watermark. In this case, the unmarked data contains two frames of 16 that contain the synchronization frame. This causes the detector to register the watermark statistic unusually high for an unmarked dataset. It is important to also recognize that although the unmarked data contains the synchronization frame, this does not result in a lower detection rate when the same sample is watermarked. In this system, when the synchronization frame is identified, the detector decodes the watermark in the next frame and advances past the watermark frame to locate the next synchronization frame. Advancing to the next frame prevents determining the presence of any additional

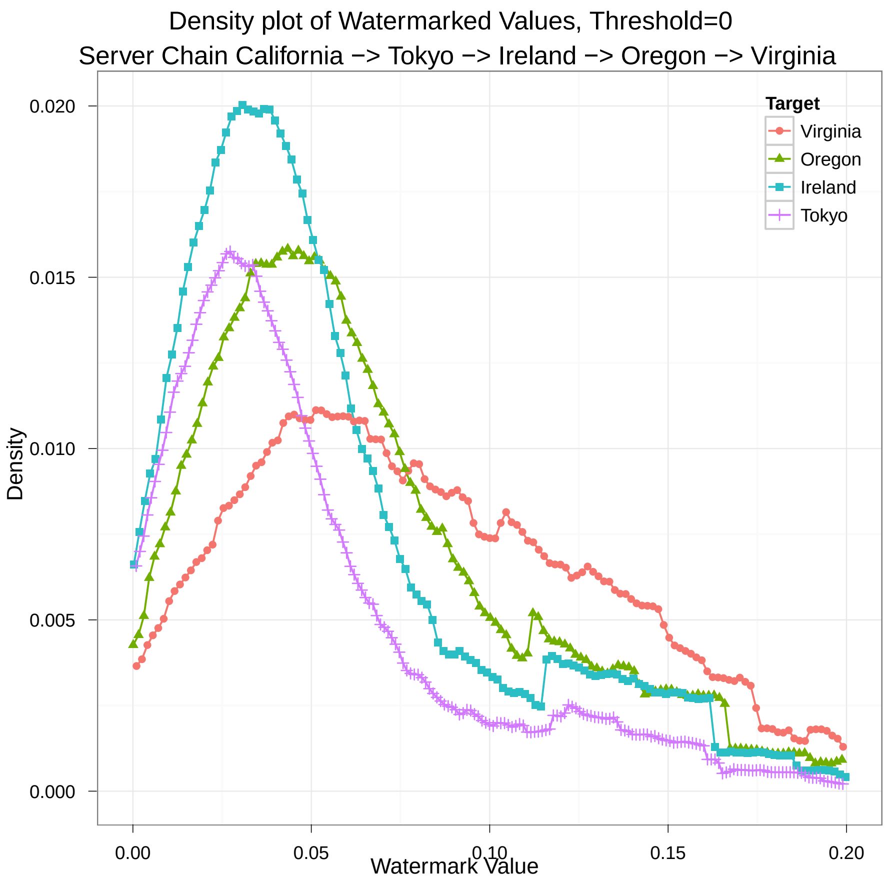


Figure 4.7: Density of watermarked sessions during live experiment.

synchronization frames inside the current window. This causes a sample such as data point 99 to appear to have a large unmarked value when the actual presence of a synchronization frame across the entire sample is only 0.01867. This flaw in the calculation of the statistic is also described in the future work along with a recommendation to correct for these unique cases.

4.5.4 Watermark Bit Errors. Another important aspect of the system occurs after the detector threshold determines that a watermark is present. The extracted values should match the code watermarked by the client in order to correlate that stream correctly. As expected, the Tokyo stream generated the greatest percentage of watermark bit errors and the Virginia server contained the least. Figure 4.8 shows the histogram of the number of bit errors in those incorrectly decoded watermarks. Based on the results that most errors were a single bit, some method of error correction in addition to the parity may enable the detection and correction of these errors in future systems.

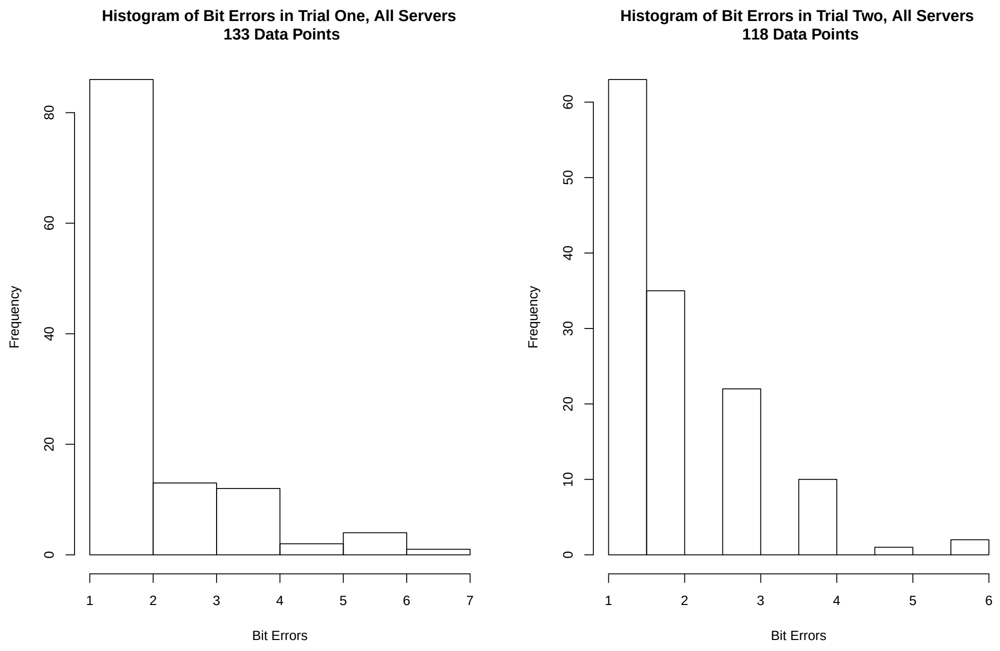


Figure 4.8: Histogram of bit errors occurring during two trials. Only watermarked streams with bit errors present are represented in this plot.

4.5.5 Wilcoxon Rank Sum Test. The Wilcoxon rank sum test is conducted using the statistic between the watermarked and unmarked data sets to measure the statistical difference. The Wilcoxon test is used because the distributions of the watermark statistics are unknown. The two analyses determine the statistical differences between the unmarked and the watermarked data and between the unmarked streams to the different servers. The p-value of the Wilcoxon rank sum test for the unmarked data against any of the marked sources is 2.2×10^{-16}. Any p-value less than 0.001 indicates a sufficient statistical dissimilarity demonstrating the difference between the unmarked data and the watermarked data sent through any of the routes in this experiment.

Although there is a statistically significant difference between any of the watermarked and unmarked datasets, the noise should not have a similar impact on the unmarked data. When an unmarked stream is sent, the route and network noise should not make the presence of a watermark any less or more detectable. A Wilcoxon rank sum test performed on the four sets of unmarked data (i.e., Tokyo, Ireland, Oregon, and Virginia) show that there is not a statistical difference between the unmarked data. Table 4.1 illustrates the p-values for each pair of tests.

Table 4.1: P-values for Wilcoxon Rank Sum test between unmarked datasets

	Ireland	Oregon	Virginia
Tokyo	0.7818	0.7288	0.09962
Ireland	X	0.4832	0.06921
Oregon	X	X	0.197

In addition to testing the difference between the unmarked and watermarked data, it is also interesting to note that there is a significant difference between some of the watermarked datasets. While the difference between sending the watermarked streams through Tokyo and Ireland is not necessarily significant ($p > 0.001$), all of the other datasets show a significant statistical difference based on the p-values ($p < 0.001$) in Table 4.2.

These results a statistical significance of the watermarked streams exists between servers. Although the original research goal is not locating the origin of the stream, the

Table 4.2: P-values for Wilcoxon Rank Sum test between watermarked datasets

	Ireland	Oregon	Virginia
Tokyo	0.1148	4.129×10^{-9}	2.2×10^{-16}
Ireland	X	1.395×10^{-6}	2.2×10^{-16}
Oregon	X	X	3.642×10^{-6}

p-values indicate a statistical difference between routes that traversed continents and those that did not.

4.5.6 Results of five trials. Four additional runs establish a statistical basis for the individual experiments. In each trial, three watermarked and one unmarked stream are simultaneously sent through the network paths described previously. Each run consists of 300 watermarked samples and 100 unmarked streams randomly chosen from the previously used CAIDA 2009 dataset of 99 samples. Table 4.3 and Figure 4.9 show the 95% confidence interval for the watermark statistic, FP and FN rates in this experiment.

The results of the FP values demonstrate that there is not a statistical significance between the target servers as shown in Figure 4.9(b). Although the interval for the Virginia is smaller than the other targets, it still falls within the interval of Ireland and Oregon confidence intervals. This smaller interval is most likely caused because this path includes no stepping stones and contains the least network noise.

Different from the FP rates, some of the FN rates are statistically different from each other. As seen in Figure 4.9(c), the Tokyo and Ireland servers along with the Oregon and Virginia targets are not statistically significant from each other; however, these two sets are significantly different in the live experiment from one another. As these two datasets are located on different continents, this provides evidence that the FN rate is determined by the watermark path.

The accuracy rate in which the detector correctly extracts the watermark is observed in five trials as well. Figure 4.9(a) demonstrates that extracted watermark accuracy is dependent on the network path. Like the FN rate, as the number of stepping stones decreases and the network path is closer to the final target, the probability of an incorrect watermark decreases. While there is not a statistical difference between the Tokyo and Ireland servers, each of the other targets demonstrate a significantly lower error rate. The additional network noise in the Tokyo and Ireland servers also increases the variance of the watermark errors, generating a wider confidence interval.

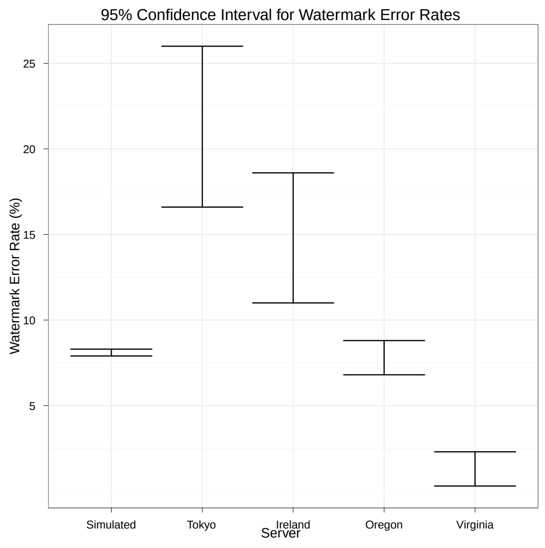
Table 4.3: Five Trial data showing 95% Confidence Intervals for the extracted watermark errors, False Positive rates and False Negative rates

	Watermark Error Rates (%)	False Positive (%)	False Negative (%)
Simulation	7.9-8.3	4.6-5.1	3.95-4.44
Tokyo	16.6-26.0	4.22-9.77	5.16-10.04
Ireland	11.0-18.6	1.22-9.57	5.11-6.52
Oregon	6.8-8.8	0.07-7.67	2.91-4.02
Virginia	0.3-2.3	3.51-4.07	0.69-3.26

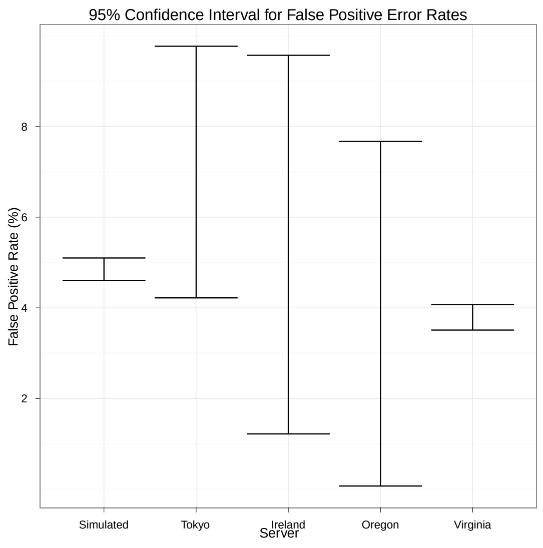
4.6 Validation

The final step of the analysis is to validate the results of the experiment. The two ways in which the data is validated is by comparing the simulated results to the live experiments and comparing the results against other previous research.

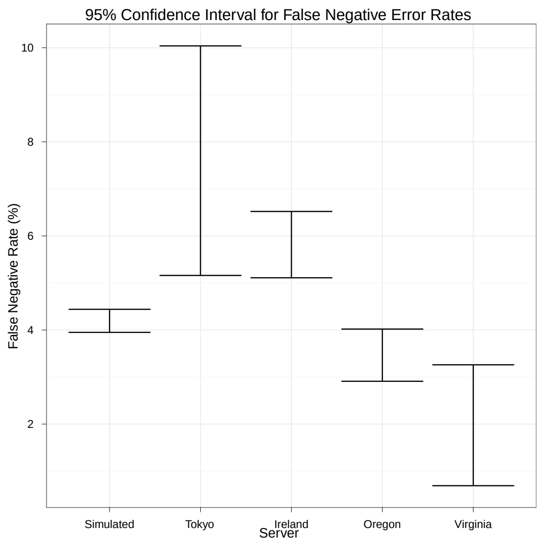
4.6.1 Comparing the Simulated and Live Experiments. The data from the live experiment closely matches the predicted data as shown in Figure 4.12. The unmarked



(a) Watermark Error Rates



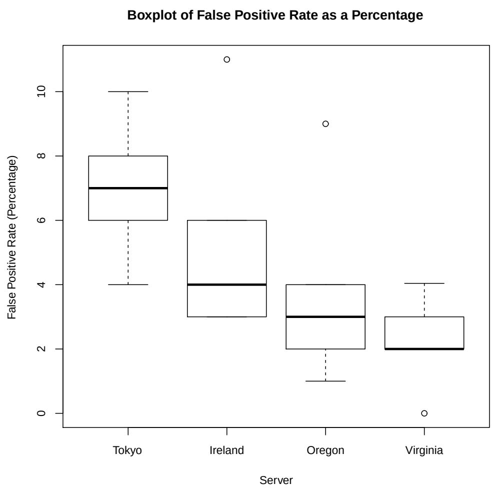
(b) False Positive Rates



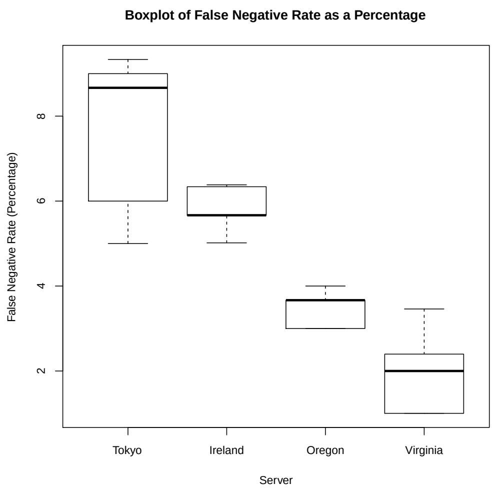
(c) False Negative Rates

Figure 4.9: Confidence Intervals for the False Positive, False Negative and Error Rates for the five trials.

and the watermarked data follow closely the density for the live and simulated experiments. The simulated data shown in black shows that the distribution of watermark statistics are a close representative of the live experiment. Even the anomaly of dataset 99



(a) Unmarked False Positive Rate



(b) Watermarked False Negative Rate

Figure 4.10: Box Plots of False Positive and False Negative rates expressed as a percentage over five trials. Each trial represents 100 data points for unmarked streams and 300 samples for watermarked streams.

referenced previously is shown both in the simulated and live datasets at a value of approximately 0.03. The validation also shows that the simulated traffic is close to the worst-case prediction of the Tokyo and Ireland servers. The traces that traversed paths within the Continental US (i.e., California, Oregon, Virginia) have a mean value greater than the predicted simulated values.

4.6.2 Detection Rates. The FP and FN rates shown in Table 4.3 also validate that the detection rates of the live experiment are statistically similar to the simulation. The FP detection rate confidence intervals all include the estimated 5% FP detection for the CAIDA dataset. Again, this demonstrates that the unmarked data results are generally independent of the target and number of stepping stones. The FN rates and the watermark extraction error rates vary more because they have greater dependence on the amount of network noise. Both of these rates are within the range of the Oregon target, but the

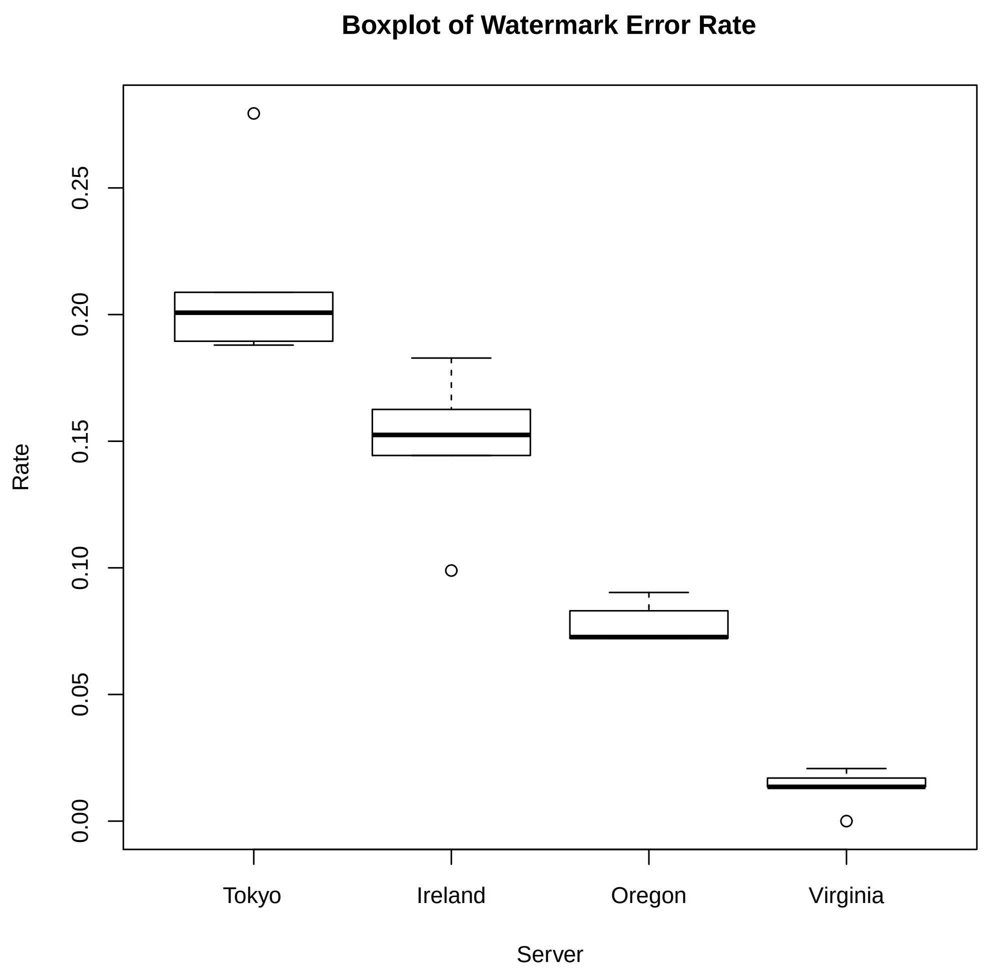


Figure 4.11: Box Plot of Watermark Error Rate of all watermarked data streams over five trials, 300 samples each trial.

Ireland and Tokyo targets show a greater FN and error rate than the simulation. This can be expected as those targets contained multiple stepping stones as well as greater distances.

4.7 Analysis Summary

The initial simulation offered an analytical basis to set the threshold to determine if the watermark is present. This value of γ is set between the mean of the quantile which

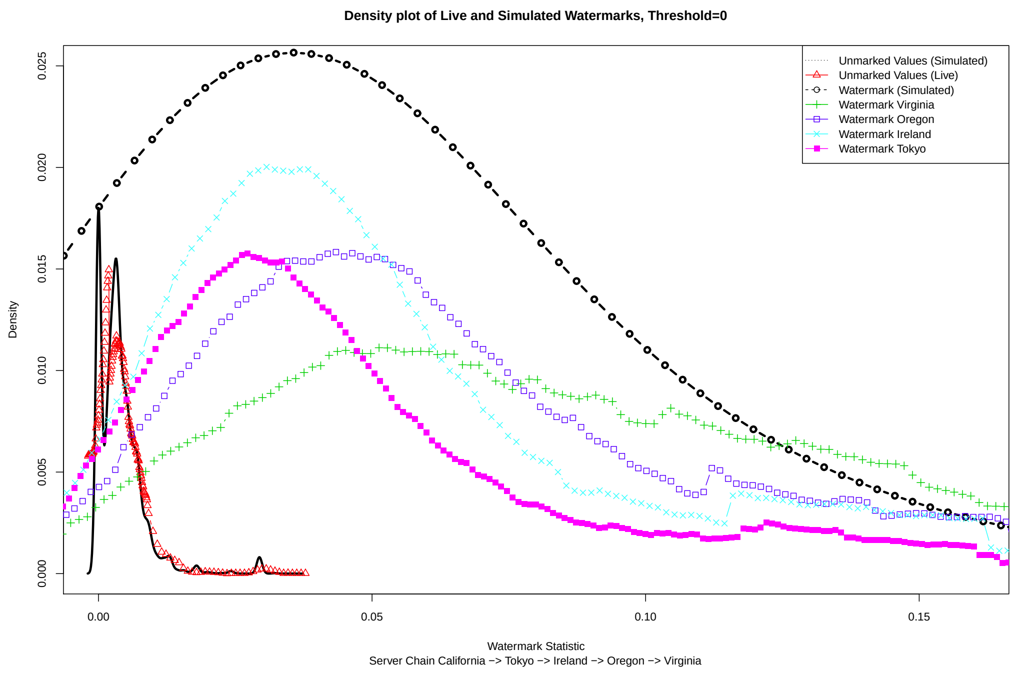
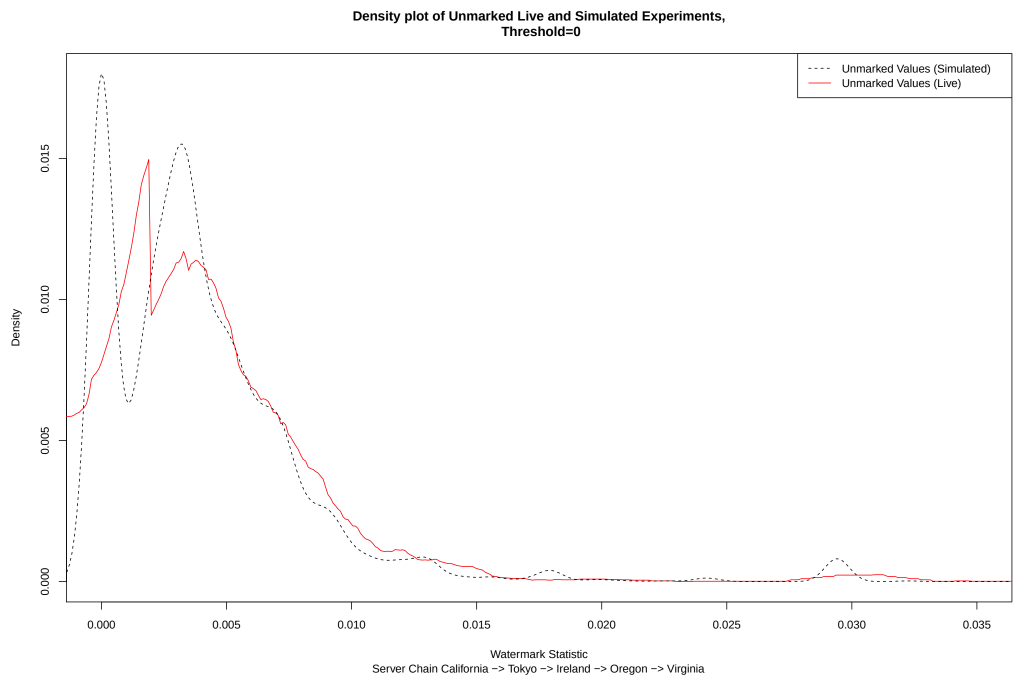


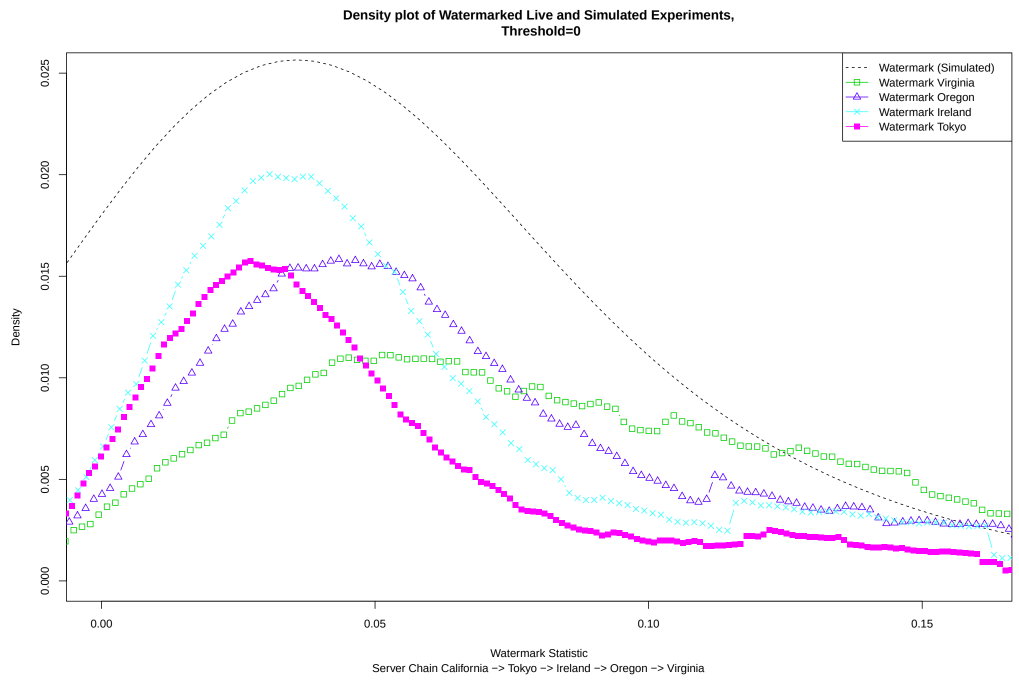
Figure 4.12: Density plot of Watermark Statistics over All Servers. Unmarked data from all servers is grouped into unmarked values.

minimized the FP and FN rates at 0.01043. The live experiment runs of 300 watermarks and 100 unmarked samples with a random watermark show very similar results to the Tokyo and Ireland network paths. As the network path progresses closer to the destination, the results show that the watermark is more discernible for the Oregon and Virginia paths. The 95% confidence interval demonstrates a statistical difference between the Tokyo and the Virginia network paths. The live experiments also show that the FP rates are between 0 and 10%. This value may be lowered by changing γ at the expense of less true watermarks detected. Currently, the FN rates are also between 0 and 10% but are dependent on the network path.

While the detection rate is lower than other watermarking techniques, this novel approach is valid, especially for human interactive network traffic in which the inter-arrival of packets shows a high degree of randomness affecting the d^1 vector. The next chapter applies aspects of this analysis to real world scenarios and offers future research ideas to improve this novel technique.



(a) Unmarked Experiments



(b) Watermarked Experiments

Figure 4.13: Density plot separating Watermarked and Unmarked Simulated and Live Datasets.

5 Conclusions

5.1 Research Impacts

As the threat of attackers increases, better methods of identifying and detecting malicious actors becomes more important. Using network stepping stones, these malicious users are able to better evade detection and penetrate deeper into networks. This research introduces a novel technique to detect network stepping stones using DWTs to embed a watermark on the network packet timing. The detector is best suited for interactive sessions normally generated by a human entering keystrokes. Although TCP packets are used in this research, because the technique presented is independent of the packet payload, it can also be extended to other protocols (i.e., User Datagram Protocol (UDP)).

The simulation results demonstrate that the system accurately detects the presence of a watermark at a 5% FP and FN rate for both the extracted timestamps as well as the empirical *tcplib* distribution. The simulated system also extracts the correct 8-bit watermark with a greater than 90% accuracy. Using sample datasets from previous watermark research, the extracted watermark accuracy increases to 99.98% accuracy.

The live experiment demonstrates repeatable results using real-world hosts as stepping stones. Five trials are conducted in which three simultaneous watermarked samples and one unmarked sample are sent 100 times at each different target. The results show that at a 95% confidence level, the unmarked samples are statistically different from each watermarked stream. This provides evidence that in a live environment the technique can detect network stepping stones.

The capability of the algorithm to function across large distances shows that it is resilient to one of the more popular use cases of the stepping stone. While previous research demonstrates effectiveness using single stepping stones, this research achieved the same capability using multiple stepping stones located in different continents. The statistical difference shown in the watermarked values between the servers allows for further investigation as to the possibility that this technique can assist in geographically locating a malicious host.

5.2 Attacker Defenses Against Algorithm

To find the stepping stones, the algorithm must be somewhat resilient against attacks. If the attacker sees that a watermark is present, they may attempt to subvert or remove the watermark altogether. Although this algorithm is subject to methods that reduce the detection rate, the difficulty of detecting the presence of the watermark makes it less likely the attacker will attempt them. Some of these attacks also assume that the attacker has complete control over the stepping stones.

As shown previously, automated traffic does not perform well in this algorithm. If the attacker cleverly devised a way to send the packets at a constant rate, the variance of the d^1 vector is not large enough and the changes by the noise overshadows the changes made to the sign. Other constant bit rate traffic (i.e., VoIP) is equally difficult to detect when tunneled through the SSH port .

Packet retransmission and packetization may reduce effectiveness with this algorithm. As stated previously, if one timestamp is lost in the frame, all values from that point to the end of the frame will most likely be decoded incorrectly. The redundancy of synchronization frames and smaller frame size (i.e., 16 values) provides some protection against this attack, but are not entirely resistant. In addition, if the stepping stones remove or add chaff packets, this also causes errors due to the missing timestamp. If the error occurs only during the watermark, all values proceeding the error will be decoded incorrectly. If the error is during the synchronization frame, the results vary depending on the bit threshold.

Inducing random delay at any point after the watermark is applied also is a successful attack on this system. Most of the utilities that retransmit packets only apply constant delay, thus as of this writing this attack needs custom written software to be successful.

Packetization also causes problems and is evident in some systems as shown during the client development. If any system re-packetizes the packets after the watermark is applied, this affects the inter-packet delay in the same way as if a packet is dropped. Because the TCP connection is an atomic process requiring acknowledgements and sequence numbers, most retransmission systems forward packets at the network layer. If the retransmission program is written as a socket, this may become a greater problem as a socket is more likely to combine smaller packets into a larger packet depending on the network speed and operating system.

5.3 Future Work

5.3.1 SSH Traces. Although the CAIDA dataset provides a wealth of data, the lack of payload presents the problem of determining whether the traffic is true SSH traffic. Crude statistical analysis determines that some of these traces are most likely not human interaction. Indeed, an ideal dataset is actual traces from an SSH server. A honeypot or captured traffic from an exercise may be beneficial to record timestamps used from human interactive sessions.

5.3.2 Algorithm Improvements. This algorithm relies on single bit parity to detect a bad watermark after finding the synchronization frame. Error detection and correction techniques such as Forward Error Correction (FEC) may enhance this algorithm to correct identified errors. The analysis shows that single bit errors are the most common. A single bit correction may drastically improve results, especially in noisy environments.

Another improvement relates to the watermark statistic calculation. If a synchronization frame is correct, this algorithm does not calculate the existence of the synchronization in the next two frames. In order to provide a more statistically correct calculation of the watermark presence, the algorithm should iteratively calculate the number of synchronization frames across the entire stream. This increases the number of calculations performed and most likely results in performance loss; however, it reduces the likelihood of the circumstance around data point 99 referenced in Section 4.5.3 from occurring.

5.3.3 Stepping Stone Geo-location. As shown in the analysis, the experiments demonstrate a statistical difference in the watermark statistic between the different servers. The additional goal of geo-locating the source may be assisted by the use of a watermark. In this case, the detector may be used to determine not only if a watermark is present, but estimating the traversed path. Future work using watermarks may assist the geo-location of the client origin using only the synchronization frames.

5.3.4 In-Line Usage. The most difficult task of using the detector against stepping stones is the ability to perform the watermarking on live data as it occurs. This may be conducted either at the kernel level on the host device or by developing an in-line watermarker that marks the packets as they traverse a network connection. In either case, the system currently needs to buffer at least 16 packets before it can apply the algorithm and send the packets. It may be entirely possible in the future to only buffer between two to four packets and use this information on the following packets, but there always exists a small number of packets to create the necessary inter-packet delays according to the d^1 vector. Future analysis may reduce the amount of packets needed in the buffer, although this delay may be one of the easiest ways for the attacker to identify the presence of a watermarking device.

5.4 Concluding Remarks

This research demonstrates a novel semi-blind active watermarking method of detecting stepping stones. The algorithm encodes the watermark on the first level detail vector d^1 DWT of the inter-packet delays. While other systems may perform better in terms of FP and FN rates, the invisibility and scalability this technique offers presents a unique advantage over similar active watermarking techniques.

Appendix A: Client and Server Development

The development of the client and server reveal many challenges regarding the design of a live watermarker. The issues documented here provide a basis for others developing similar or future systems that require similar functions. While some of these problems are platform dependent, many of the lessons illustrated here serve as a starting point to a full scale live watermarker.

A.1 Client Development

The client development changed courses based on a few observations during the experiment. The problems included socket streams issues, transmission delays and low level TCP programming development. The final client resolved these issues in a way suitable for experimentation and analysis, but not for a finished product. To design a product that is capable of manipulating the packet timings needs more effort and expansion. This client only served to communicate with the designed server and send TCP packets at precise time intervals.

Initially, the client used Python Socket library to generate the packets at specified time intervals. The Python socket library is robust at handling TCP sockets and provides a very simple means to send the packets from a source to destination. However, with the simplicity came problems as well. Because the socket is a stream oriented protocol, the simple client packetized the data intended for separate TCP packets. Using the server to record the timestamps did not reveal the problem because the server differentiates between the packets by a sequence sent at the end of each packet. While the server showed the timestamps correctly, the wireshark analysis critical to the experiment showed that the client packetization reduced the total number of packets by as much as 20%. Because the client repacketized small packets sent at high rates, if the packet data was larger, it may not be able to repacketize the data and force the client to send the packet individually.

Using packet sizes of approximately 1050-1100 bytes of TCP payload successfully forced the client to send each packet individually and prevented initial repacketization. This introduced the problem of transmission delay. Because each packet is now larger, it takes more time to actually transmit the message on the physical medium. Depending on the host network speed, the observed transmission delay ranged from 100 to 200 msec. Because of this delay, the only way to reproduce the necessary packet timings was to add this transmission delay to the original delay for each packet. This technique may be later used in a store-and-forward type of implementation but was not the original intent for the experiment.

The final client uses the Python Scapy module to generate and transmit packets at the link layer. The previous program correctly reads the delay times and sends the packets, but the scapy module allows for fine control over the interval between individual packets. Using scapy, the client first generates a three way handshake necessary for TCP and continues to send data at specified time intervals directly to the network interface. Because this experiment is not concerned with the receipt of the server echo replies, this was removed from the server. This simplified the client because the TCP sequence numbers only needed to be controlled from the sender. Using this new implementation, the client now sends the data at prescribed intervals and does not exhibit the packetization problems described earlier. The sender needs to have an additional command to prevent the kernel from closing the TCP connection. Because the kernel is not actually initiating the connection, when the low level packets are sent out, the server sends an acknowledgement of each packet. When the kernel sees the acknowledgement, it will send a reset to close the connection because the kernel did not initiate it. The following command prevents a TCP reset command from leaving the client, although the kernel still attempts to send the packet:

```
sudo iptables -A OUTPUT -p tcp --tcp-flags RST, RST
```

```
    --destination-port 5000 -j DROP
```

Using scapy to send the packets, the wireshark analysis shows that the client is sending each individual packet correctly and the server is also receiving each individual TCP packet separately.

A.2 Server Development

The server responsible for collecting the timestamps needs to have the capability to service multiple connections in a timely manner. The server does not necessarily need to reply to the client, but does need to acknowledge the packets as they arrive and create new sockets to discern the different connections. The Twisted Python module creates an easy manner in which this can be done in a multi-threaded fashion to optimize performance. Each connection generates a new thread handled by the server. Because the intermediate clients change the original TCP socket from the client, there is a need for the server to parse the data from the client to determine whether a watermark is sent in the experiment. In this experiment, the Twisted server searches the TCP payload and records the connection and watermark to a file for later use. For unmarked data, the client sends 9999 9999 as the watermark denoting that there is none present.

Although the original client records the timestamps as part of the Python program, this feature does not accurately measure the timestamps as they arrive to the server. Instead, tcpdump is run in a separate process in order to capture the traffic as it flows to the server. This creates the scenario in which the detector may be placed external to the server and the dumped traffic may be analyzed using an external resource.

A.3 Intermediate Clients

The intermediate clients need to pass the data from one server to the next in order to facilitate the daisy-chain operation of the stepping stones. Initially, the netcat program seemed to be an acceptable solution as a simple network relay. Observing the packet timestamps from the server showed that the packets were arriving at the expected times, but the wireshark timing showed otherwise. The main problem as seen previously is the treatment of TCP packets versus sockets. In some cases, netcat would packetize the smaller packets based on the TCP stream in to a larger packet which results in the loss of the timestamp at the network packet layer.

The next solution focused on socat. This is a powerful tool capable of relaying packets at many layers, including UNIX sockets. The following command relays the TCP packets and did not show any packetization problems evident when using netcat:

```
sudo nohup socat TCP-LISTEN:5000,nodelay,fork
    TCP:next_relay_server:5000,nodelay
```

The only issue that socat showed is the duplication of the TCP source port. Socat appears to use port 33800 as the initial source port in the TCP transmissions. Because of this, each time the intermediate server immediately preceding the destination restarted, the TCP source port restarted to 33800. This introduces problems in the analyzer because the timestamps are based on the streams that are identified based of the tuple of source IP, source port, destination IP and destination port. Section A.2 also mentions that this problem is overcome by embedding the watermark information in the TCP payload.

Appendix B: Removed Samples

B.1 Non-Random Samples

The decision to remove samples is purely a subjective one based on the fact that there is no information regarding the data payload to determine whether the traffic is true SSH traffic. Other ports may conceal traffic on port 22, as well as SCP regularly copies files on this port. This anomalous traffic removal is a manual process.

Most of the samples are removed based on two factors. First, many of the samples do not appear random and appear more like traffic generators that have constant time differentials. Examples of this traffic may include VoIP traffic, periodic maintenance and probing traffic such as Simple Network Monitoring Protocol (SNMP). The figures below represent the 180 streams removed for these reasons.

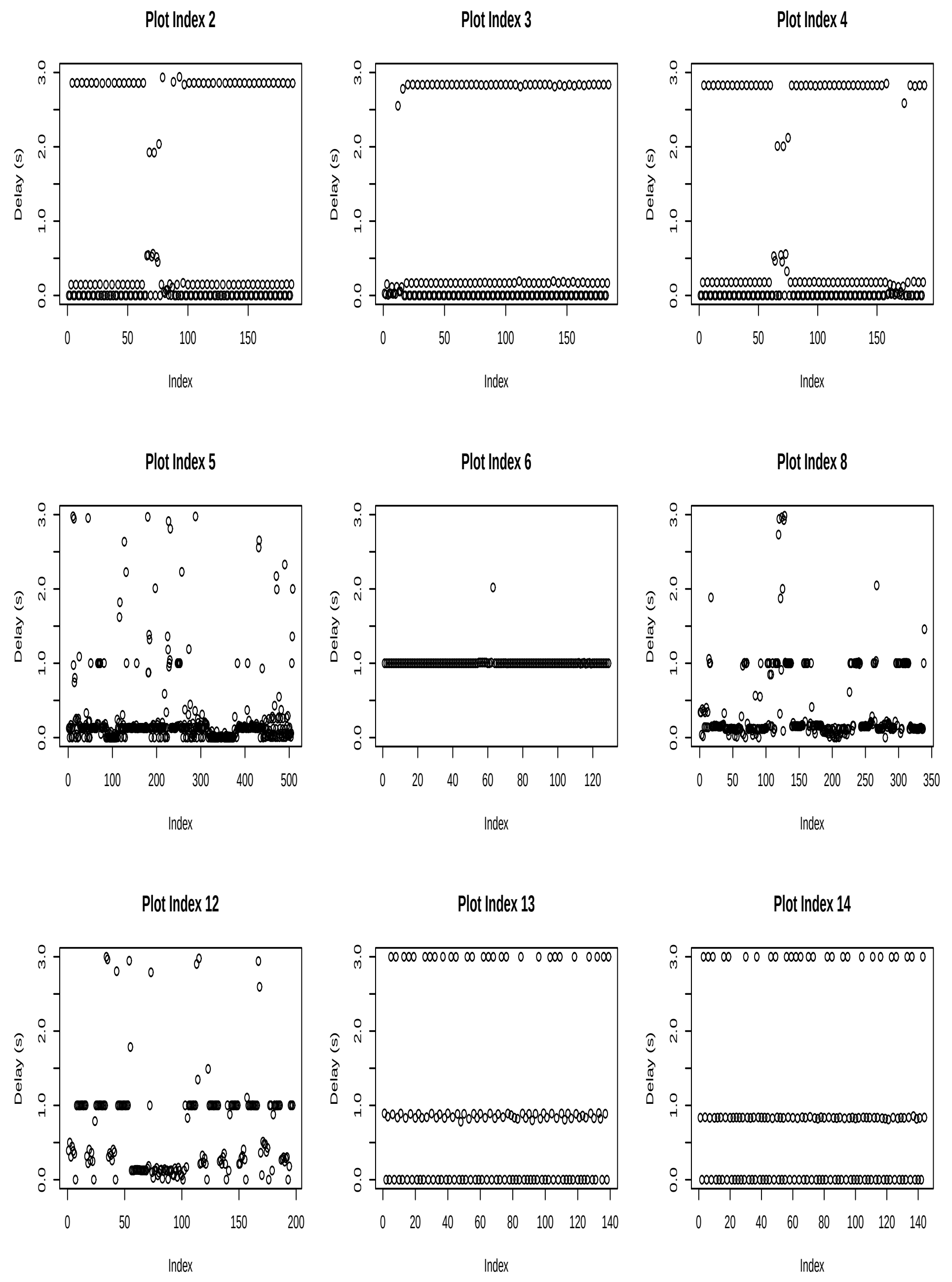


Figure B.1: Plot of Erroneous Port 22 Times removed from experiment

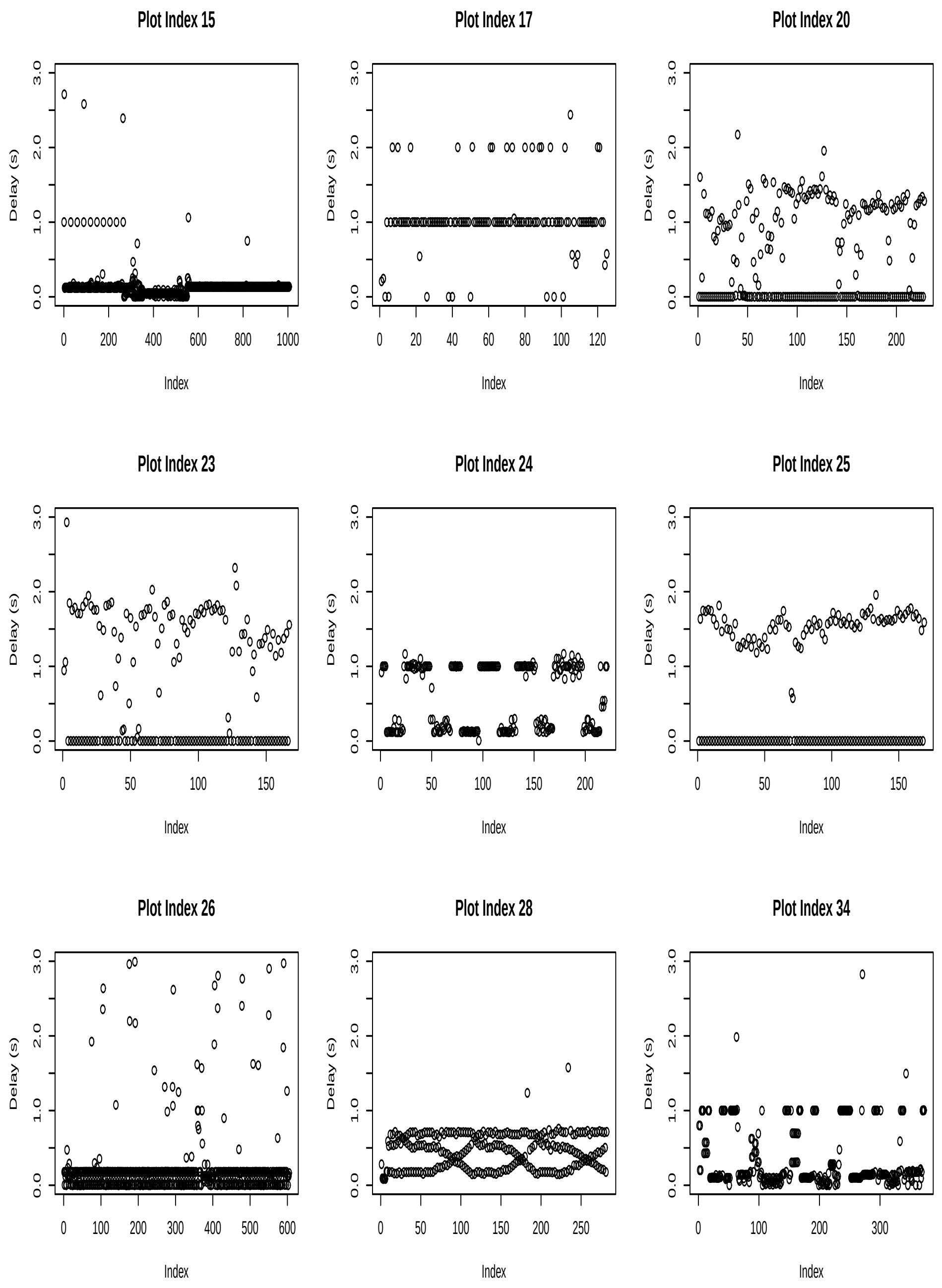


Figure B.2: Plot of Erroneous Port 22 Times removed from experiment

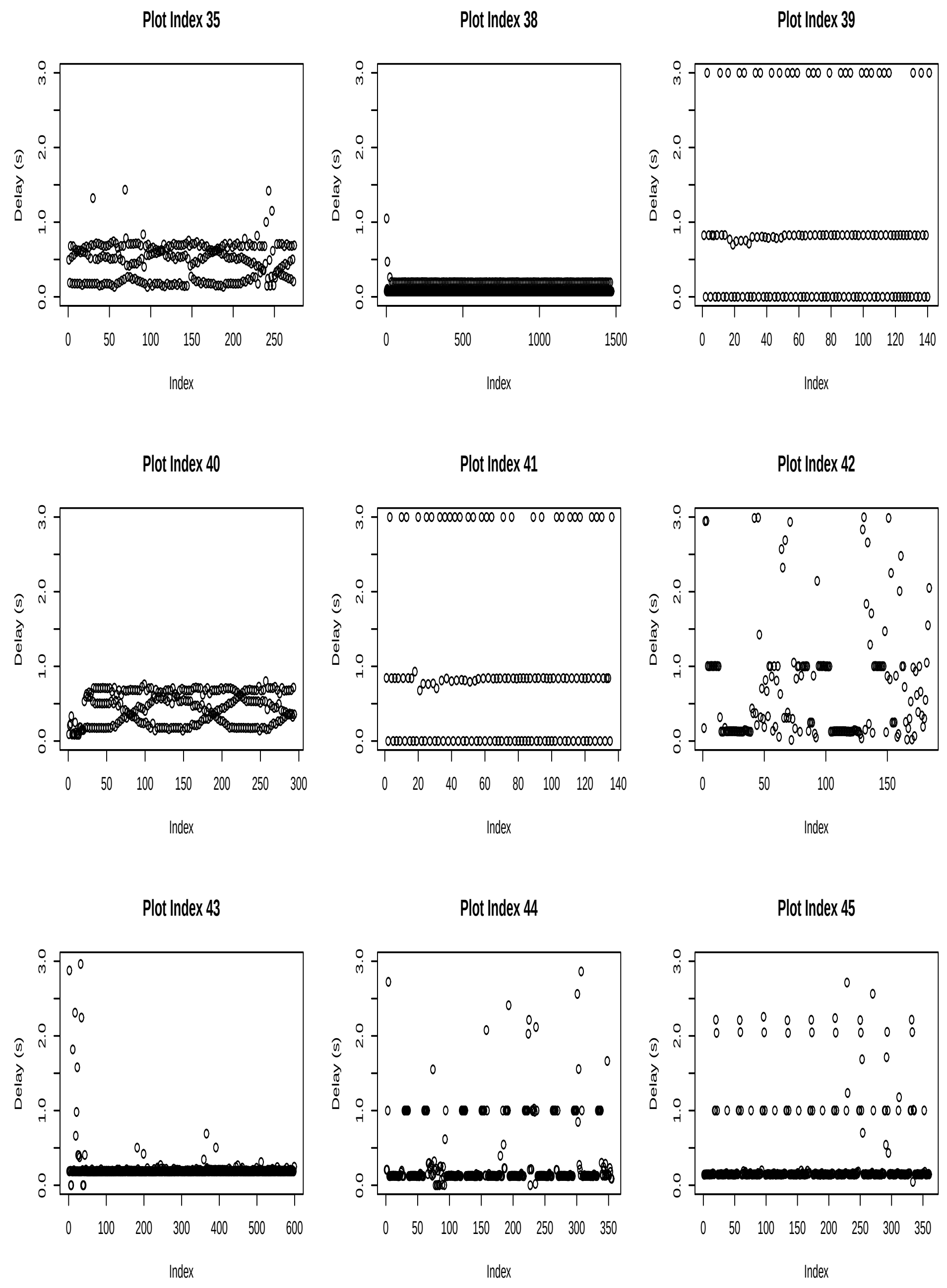


Figure B.3: Plot of Erroneous Port 22 Times removed from experiment

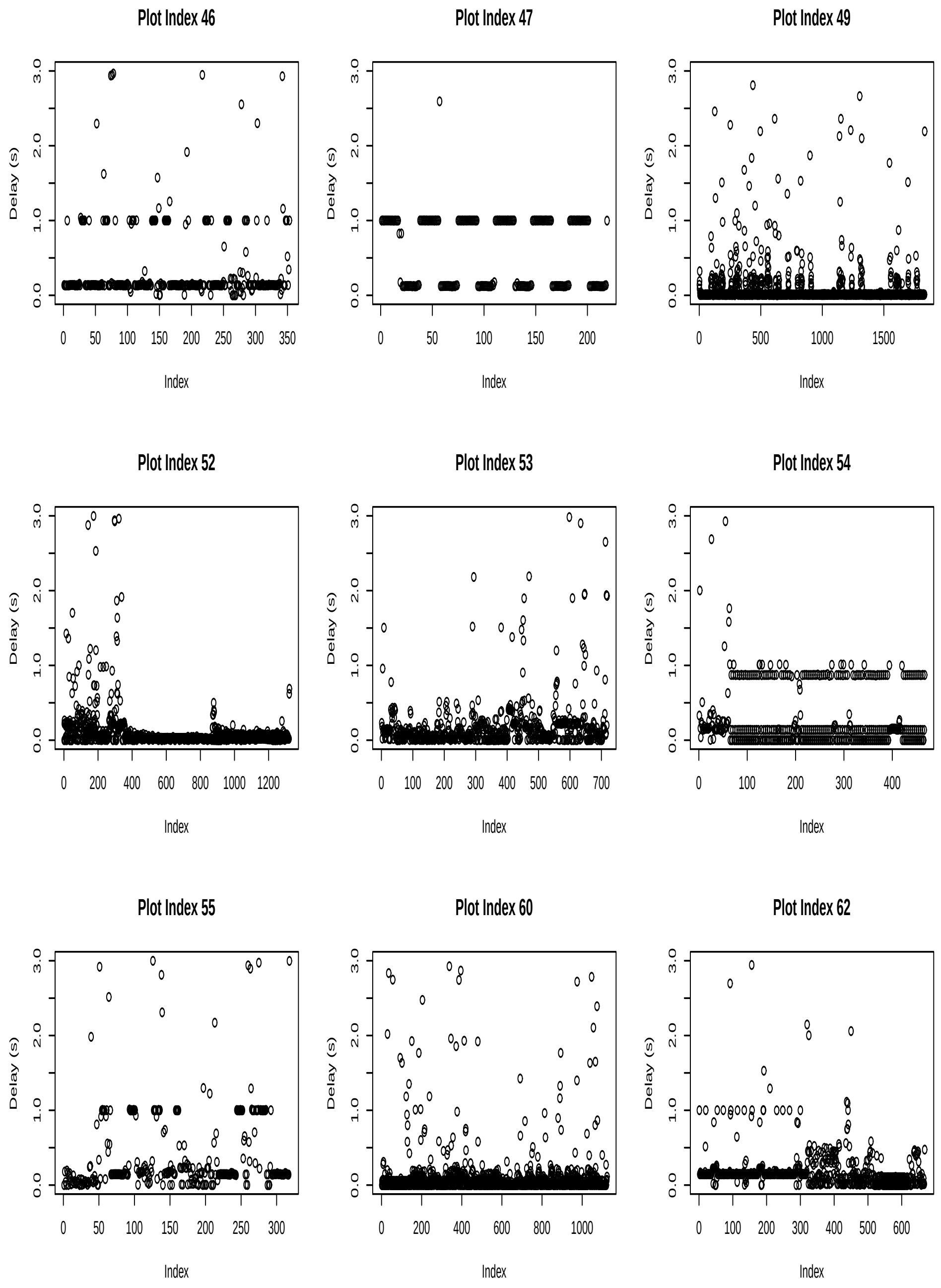


Figure B.4: Plot of Erroneous Port 22 Times removed from experiment

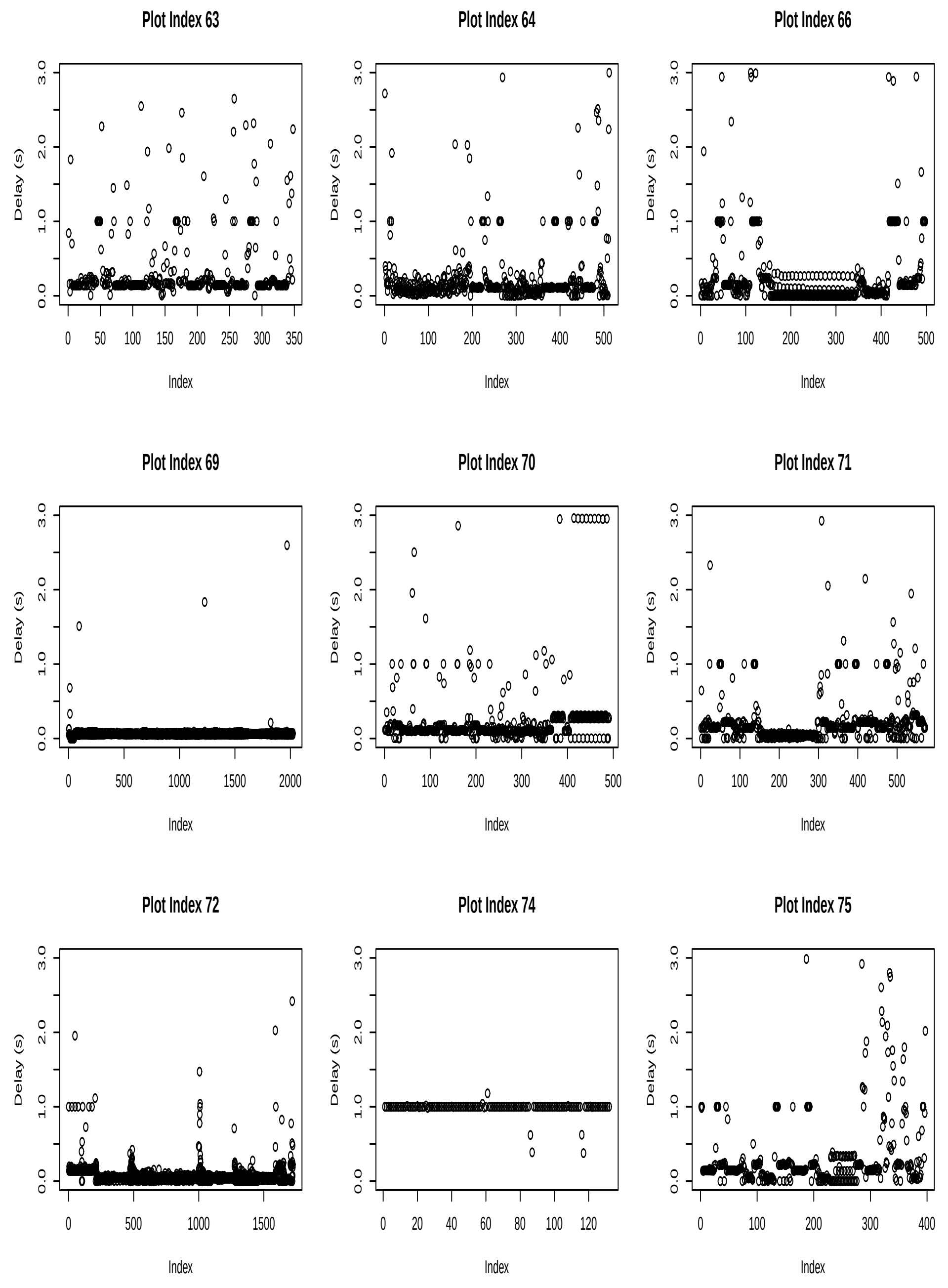


Figure B.5: Plot of Erroneous Port 22 Times removed from experiment

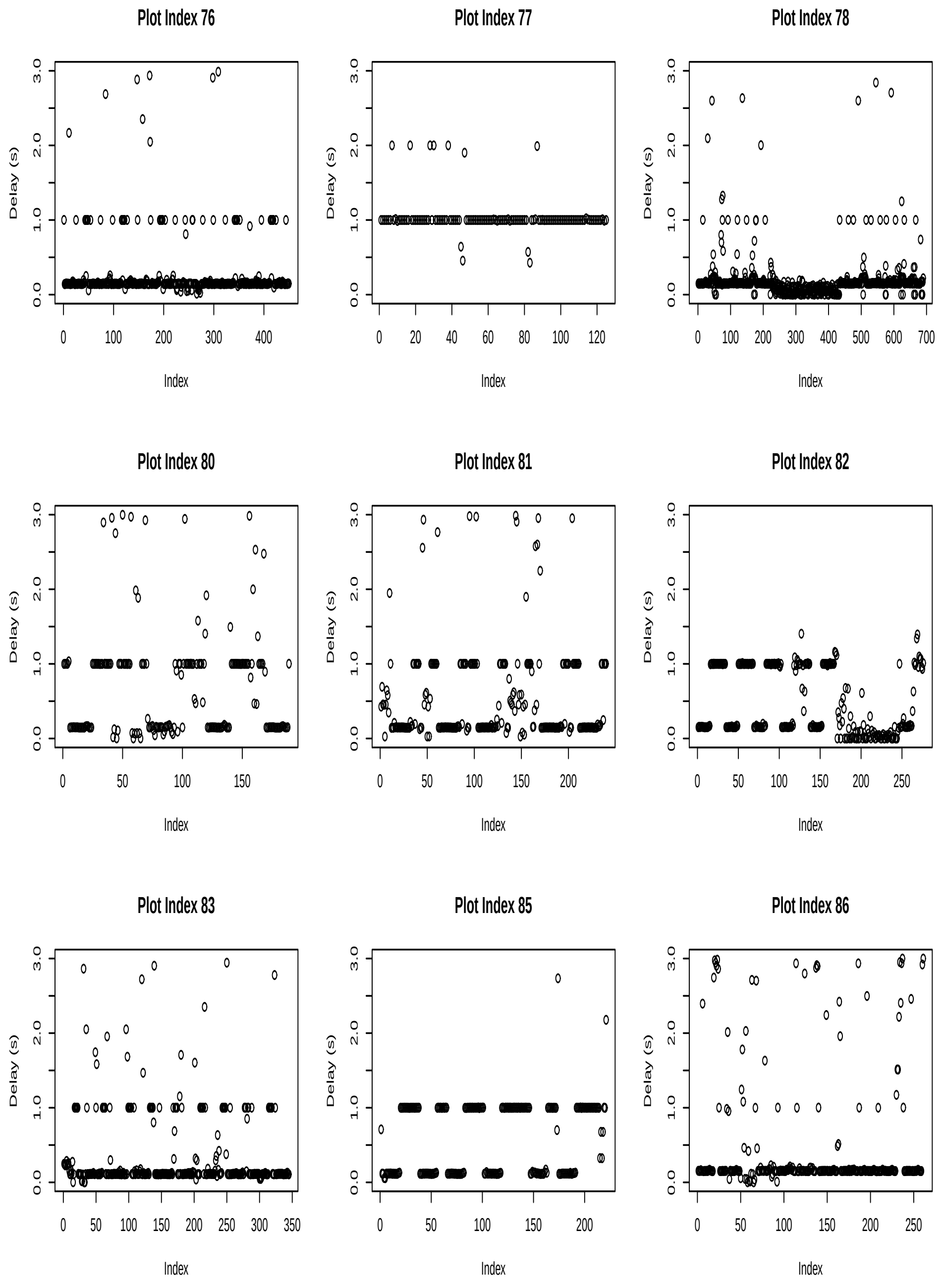


Figure B.6: Plot of Erroneous Port 22 Times removed from experiment

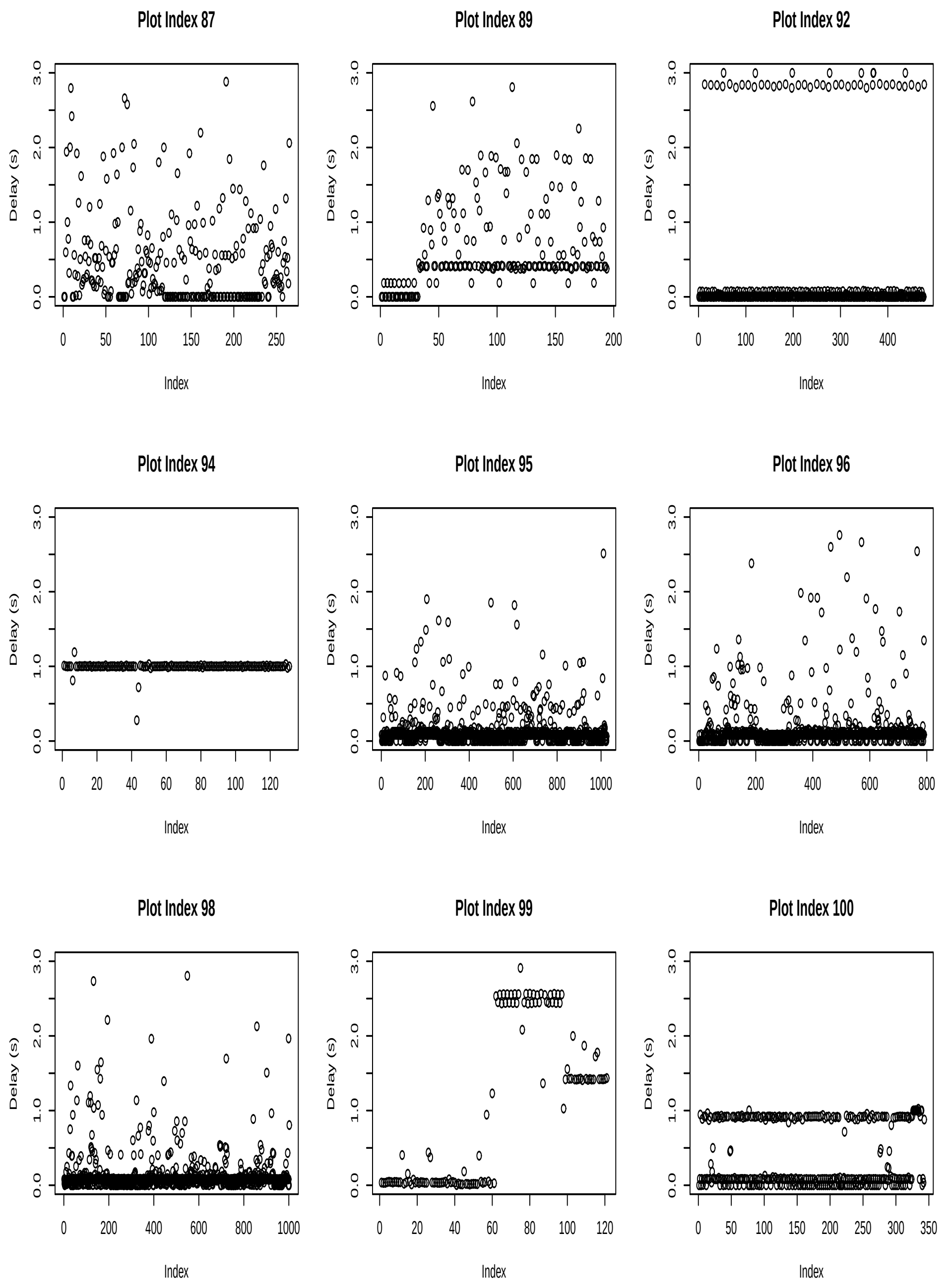


Figure B.7: Plot of Erroneous Port 22 Times removed from experiment

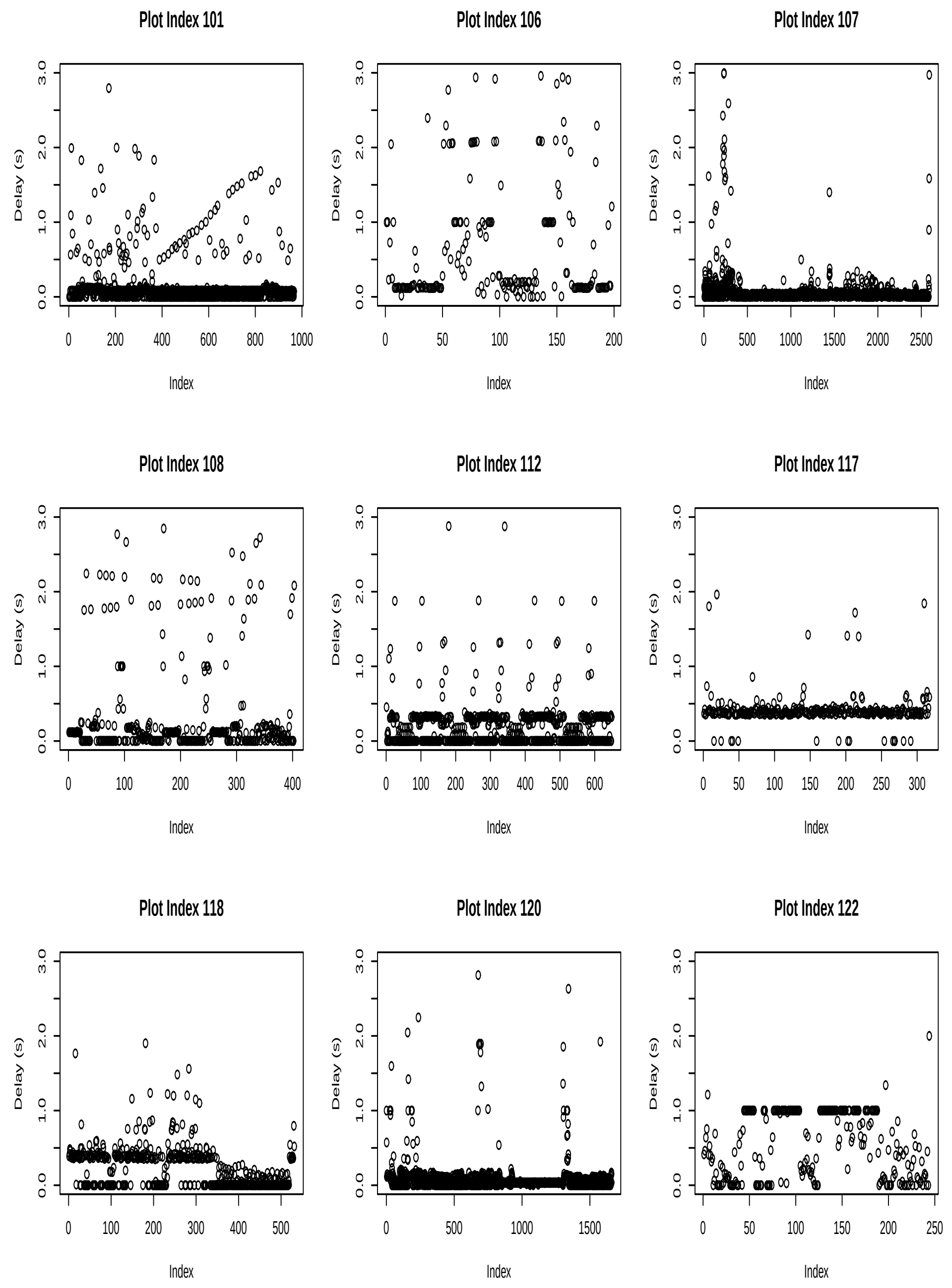


Figure B.8: Plot of Erroneous Port 22 Times removed from experiment

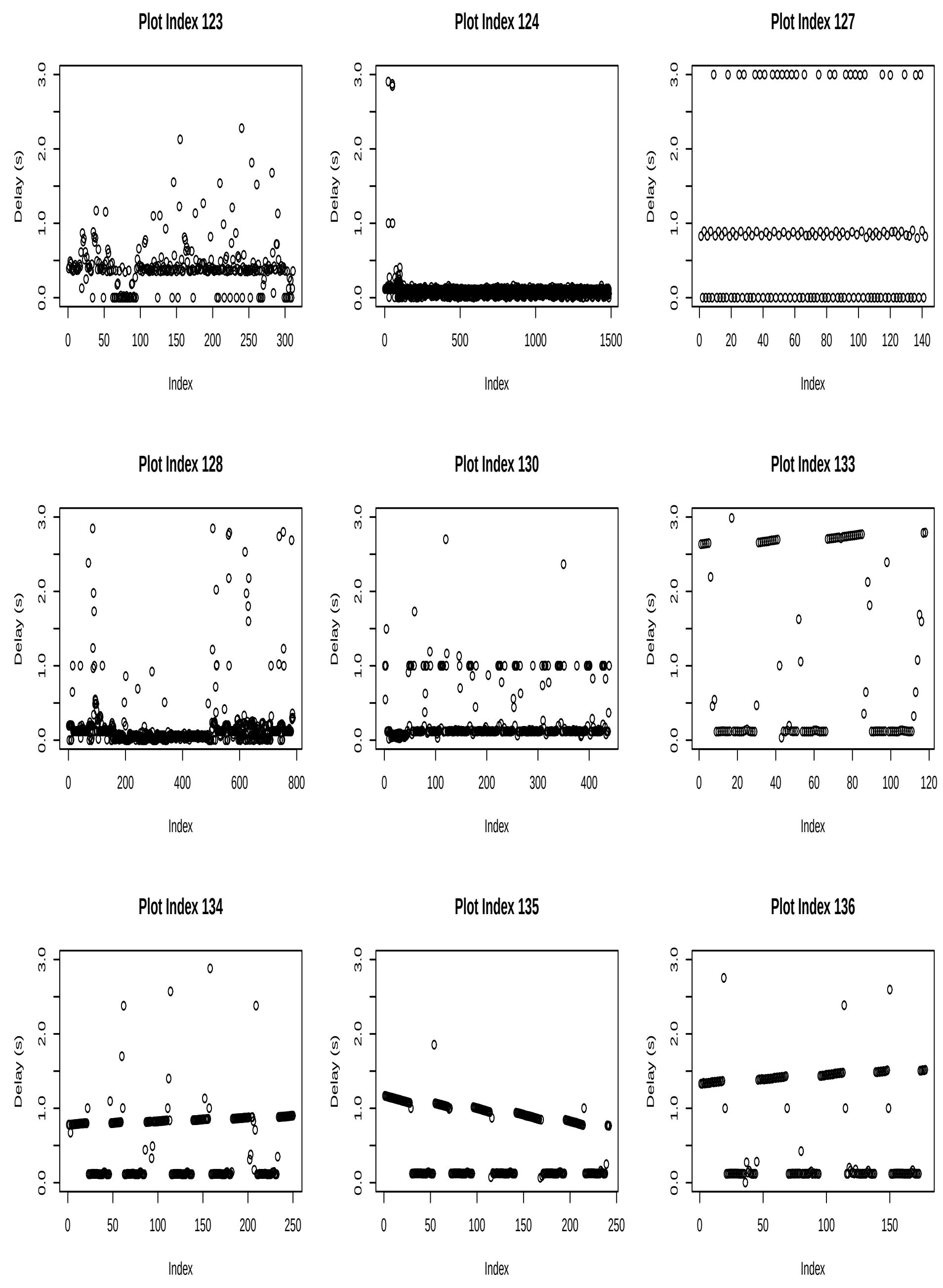


Figure B.9: Plot of Erroneous Port 22 Times removed from experiment

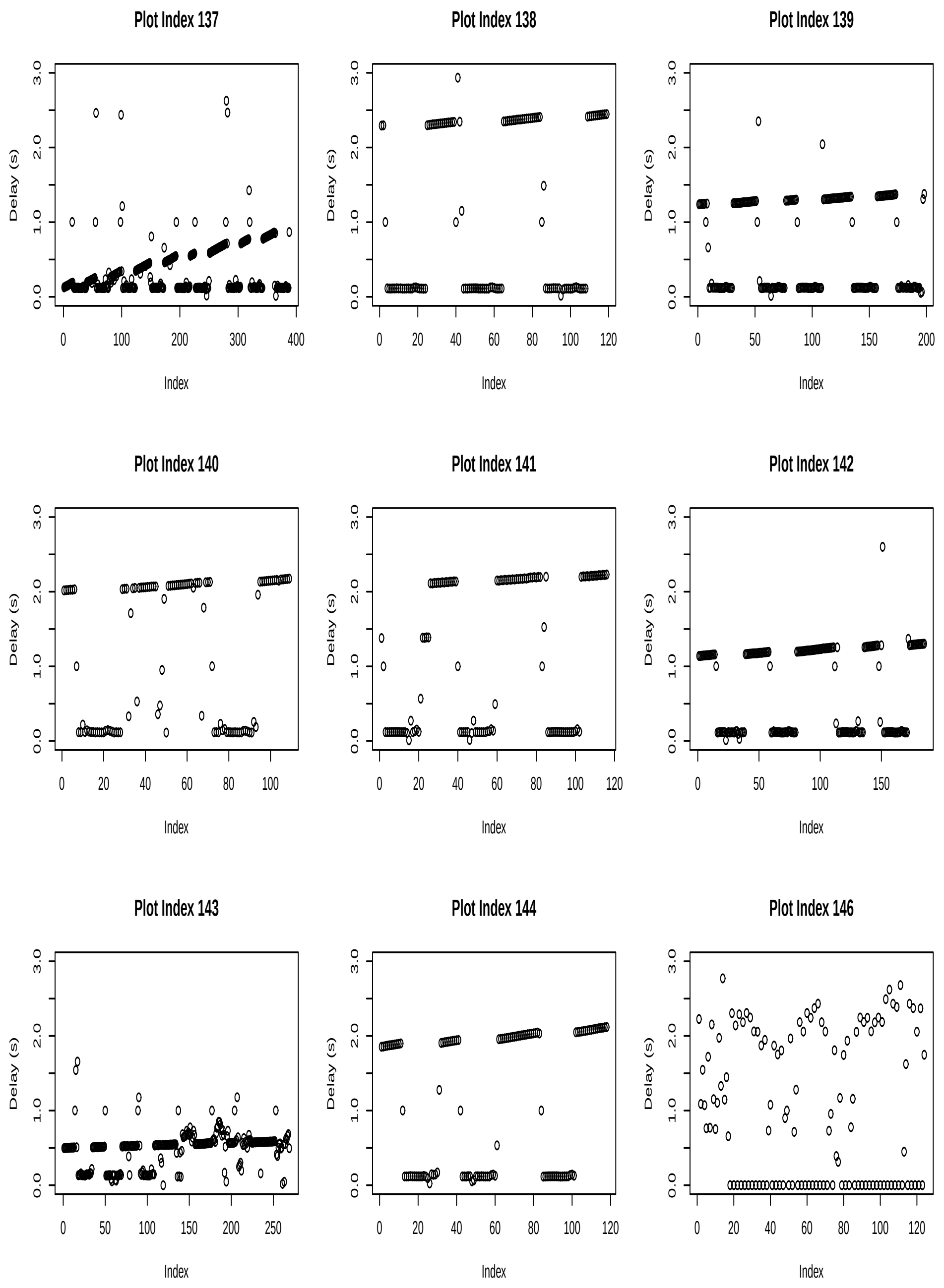


Figure B.10: Plot of Erroneous Port 22 Times removed from experiment

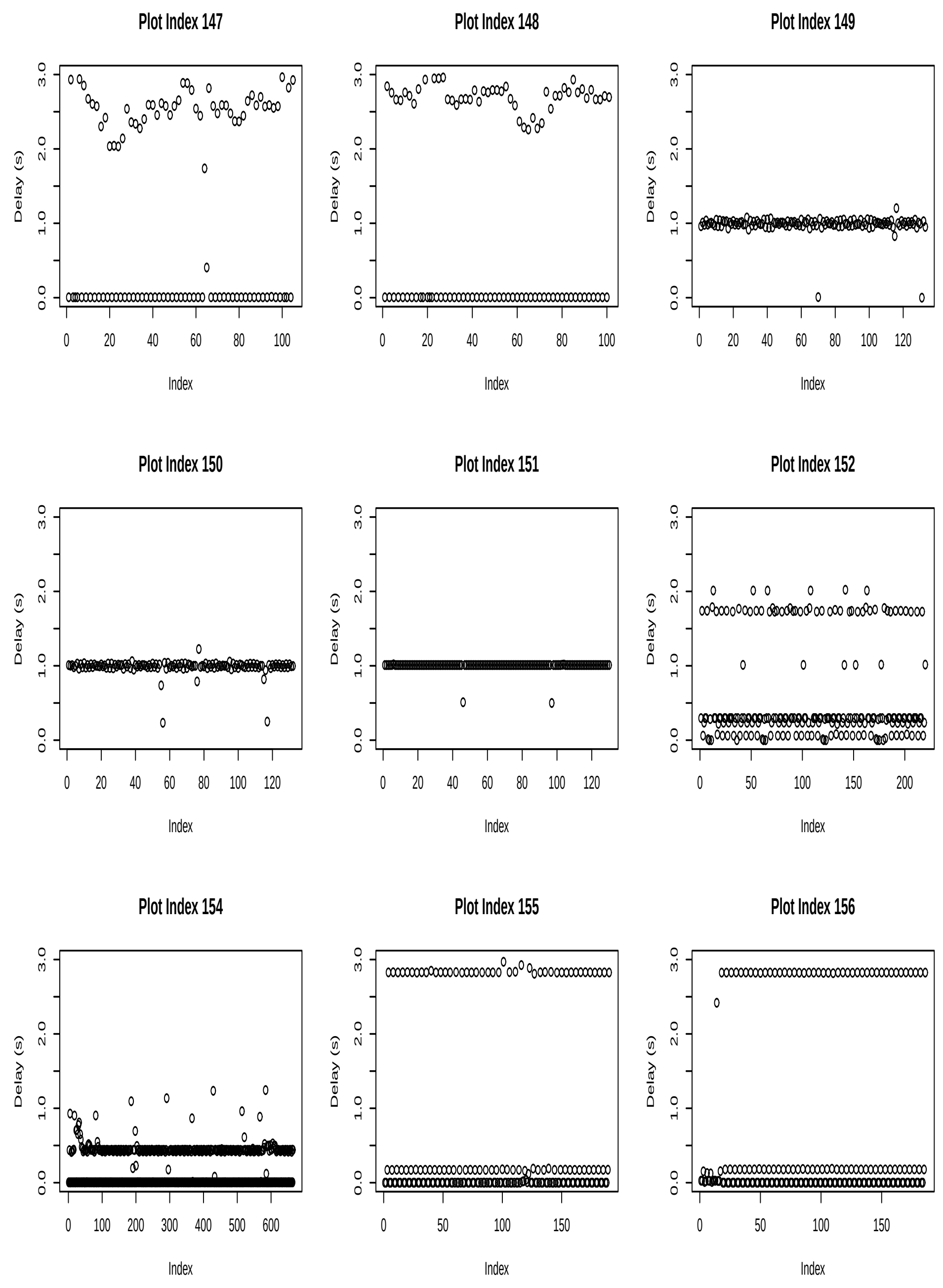


Figure B.11: Plot of Erroneous Port 22 Times removed from experiment

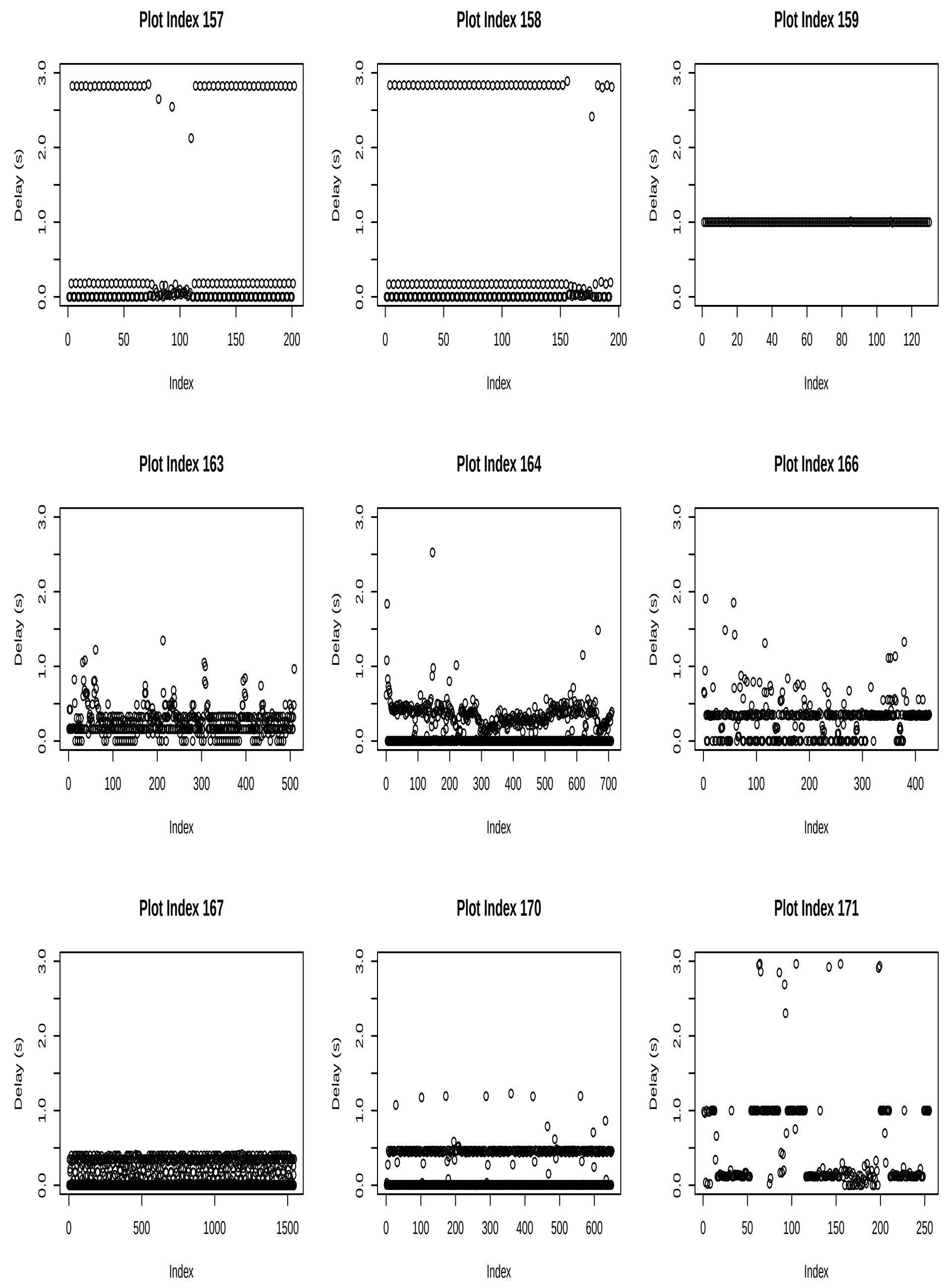


Figure B.12: Plot of Erroneous Port 22 Times removed from experiment

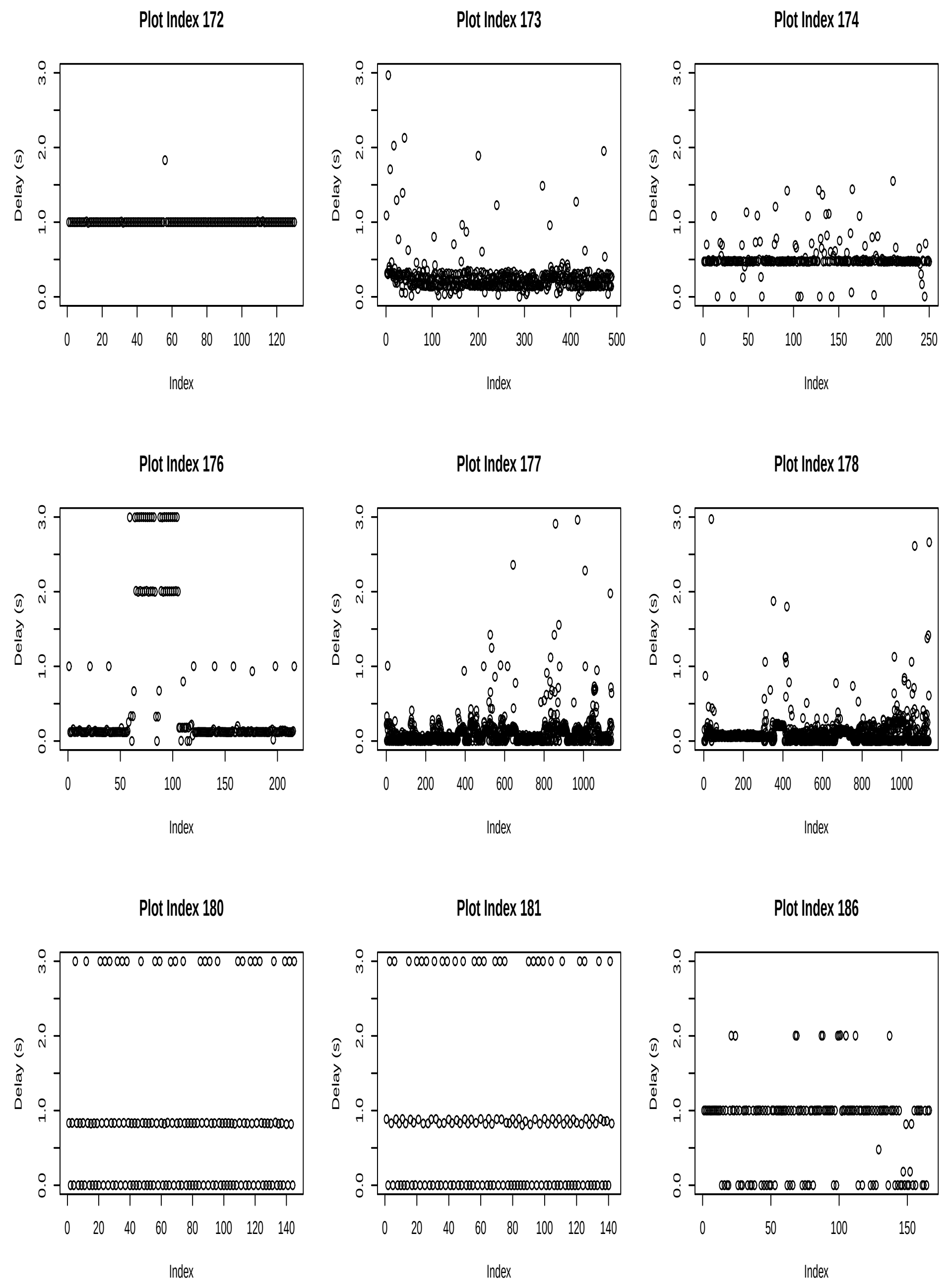


Figure B.13: Plot of Erroneous Port 22 Times removed from experiment

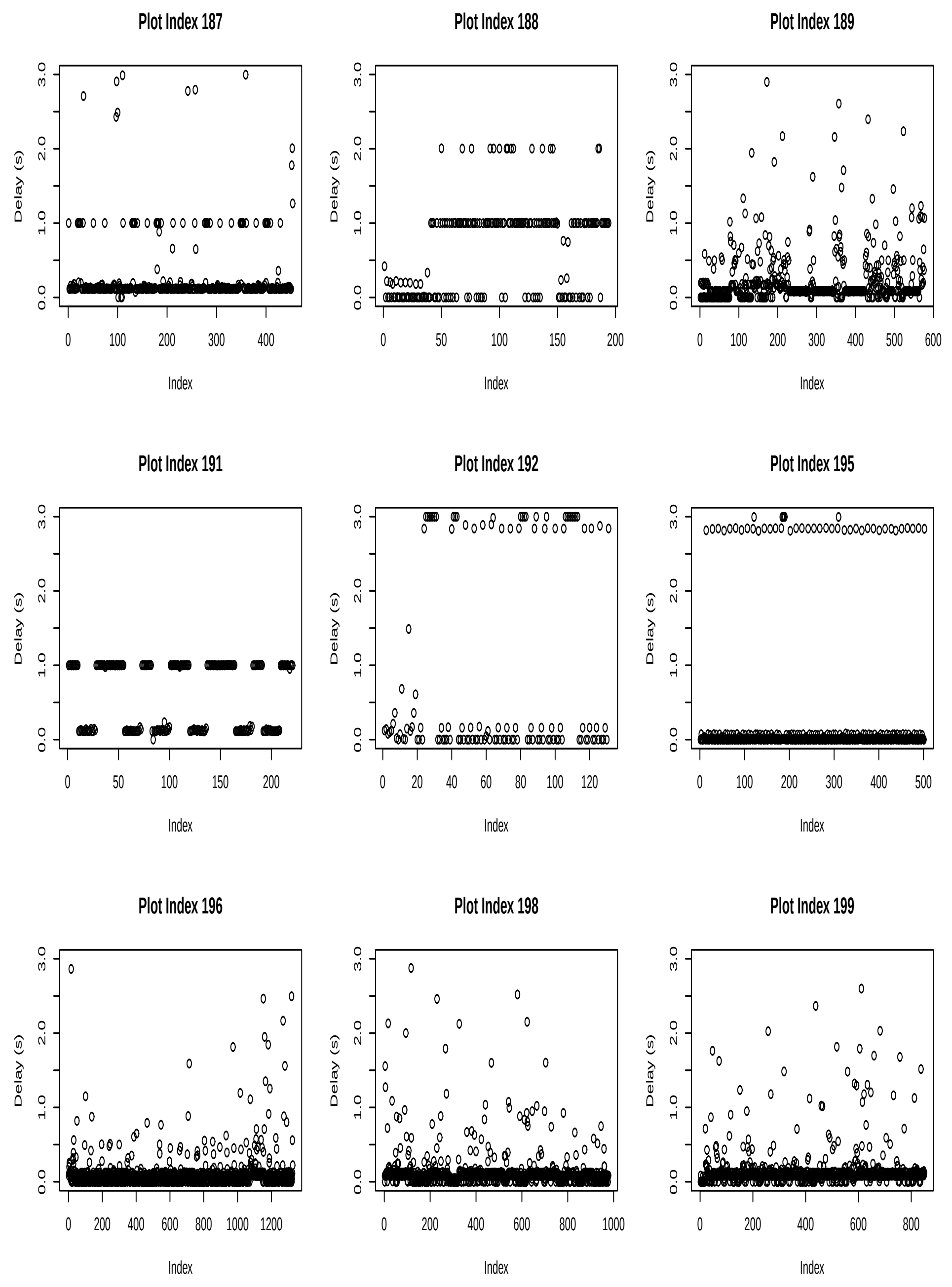


Figure B.14: Plot of Erroneous Port 22 Times removed from experiment

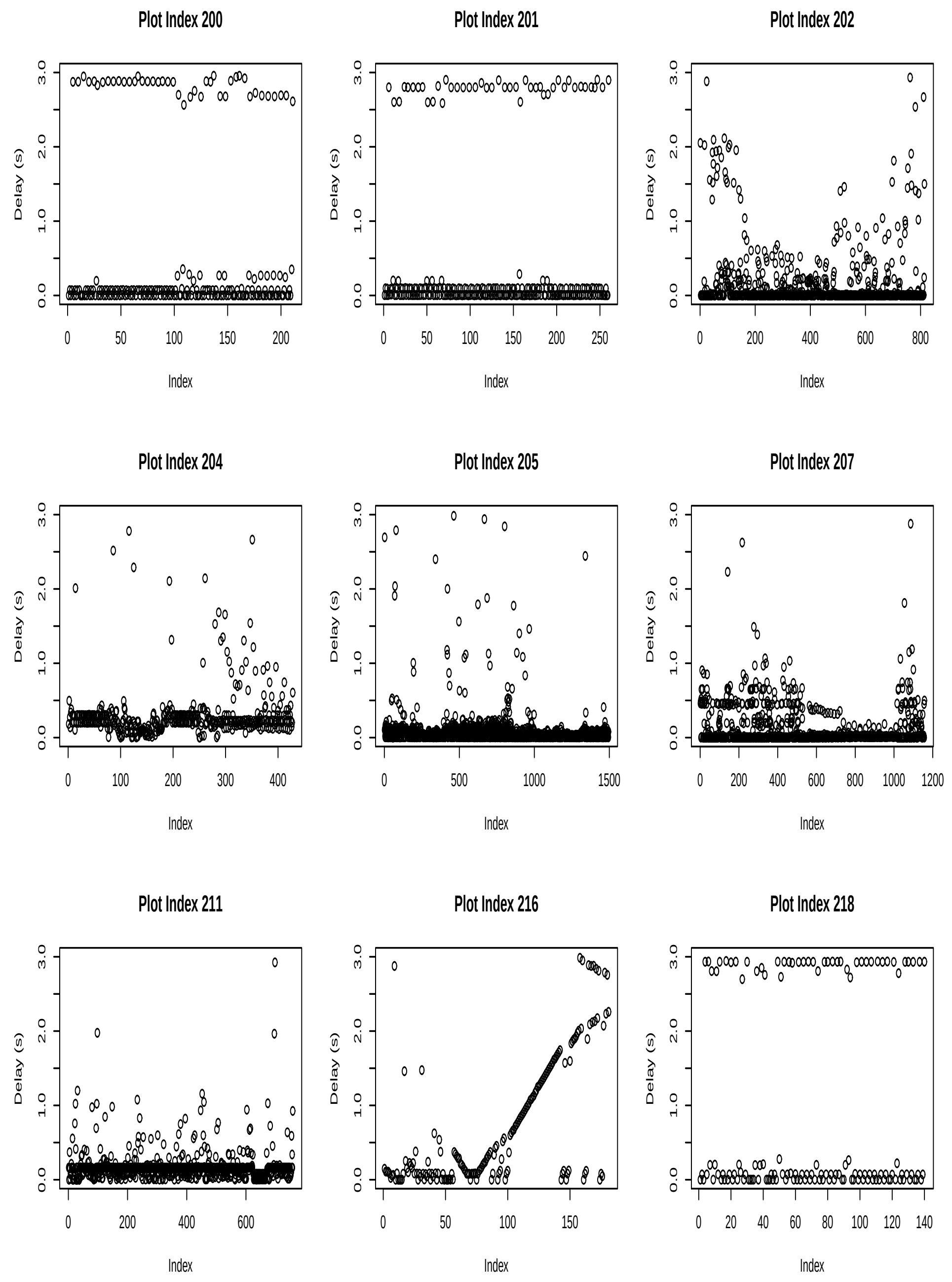


Figure B.15: Plot of Erroneous Port 22 Times removed from experiment

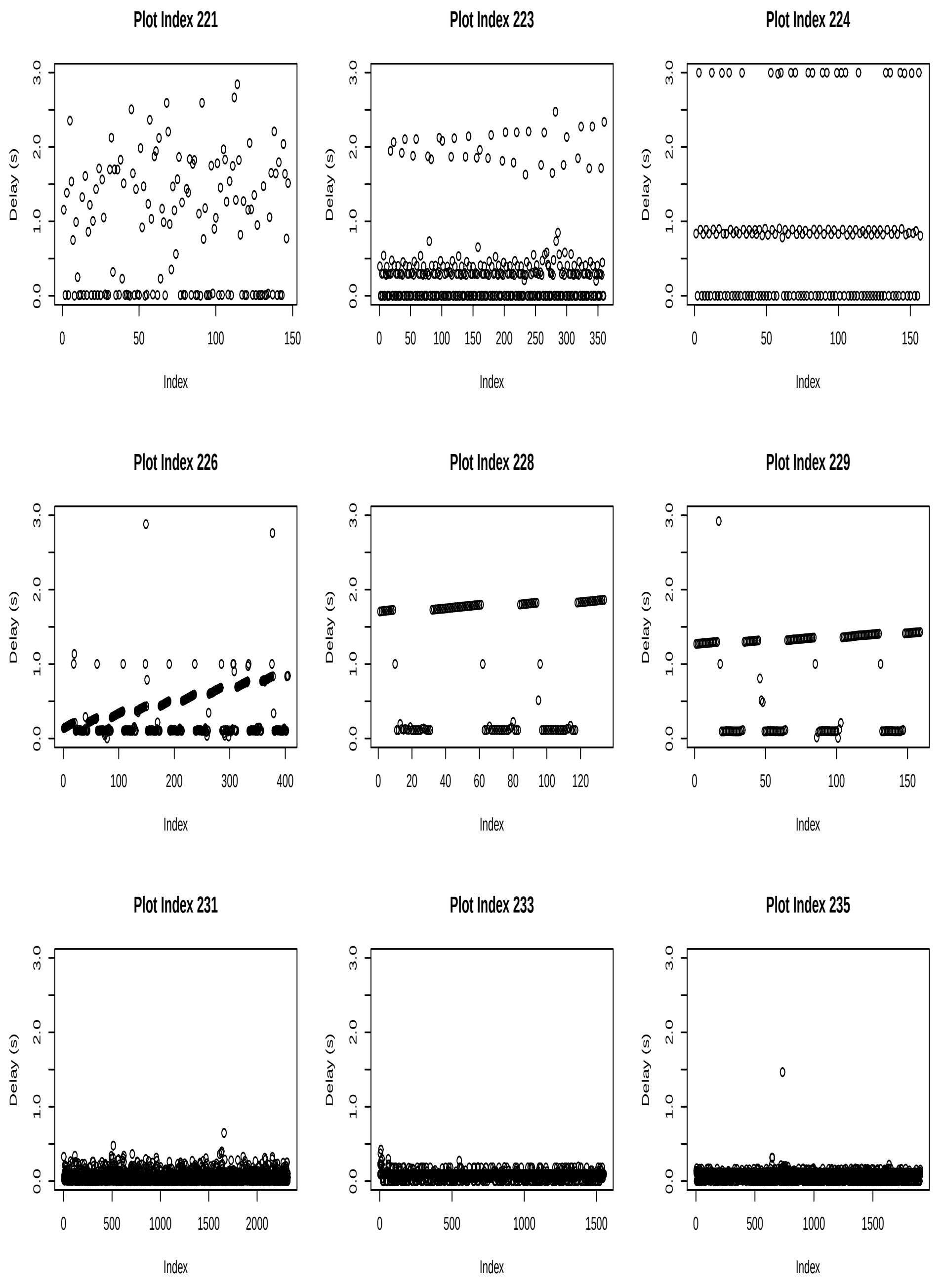


Figure B.16: Plot of Erroneous Port 22 Times removed from experiment

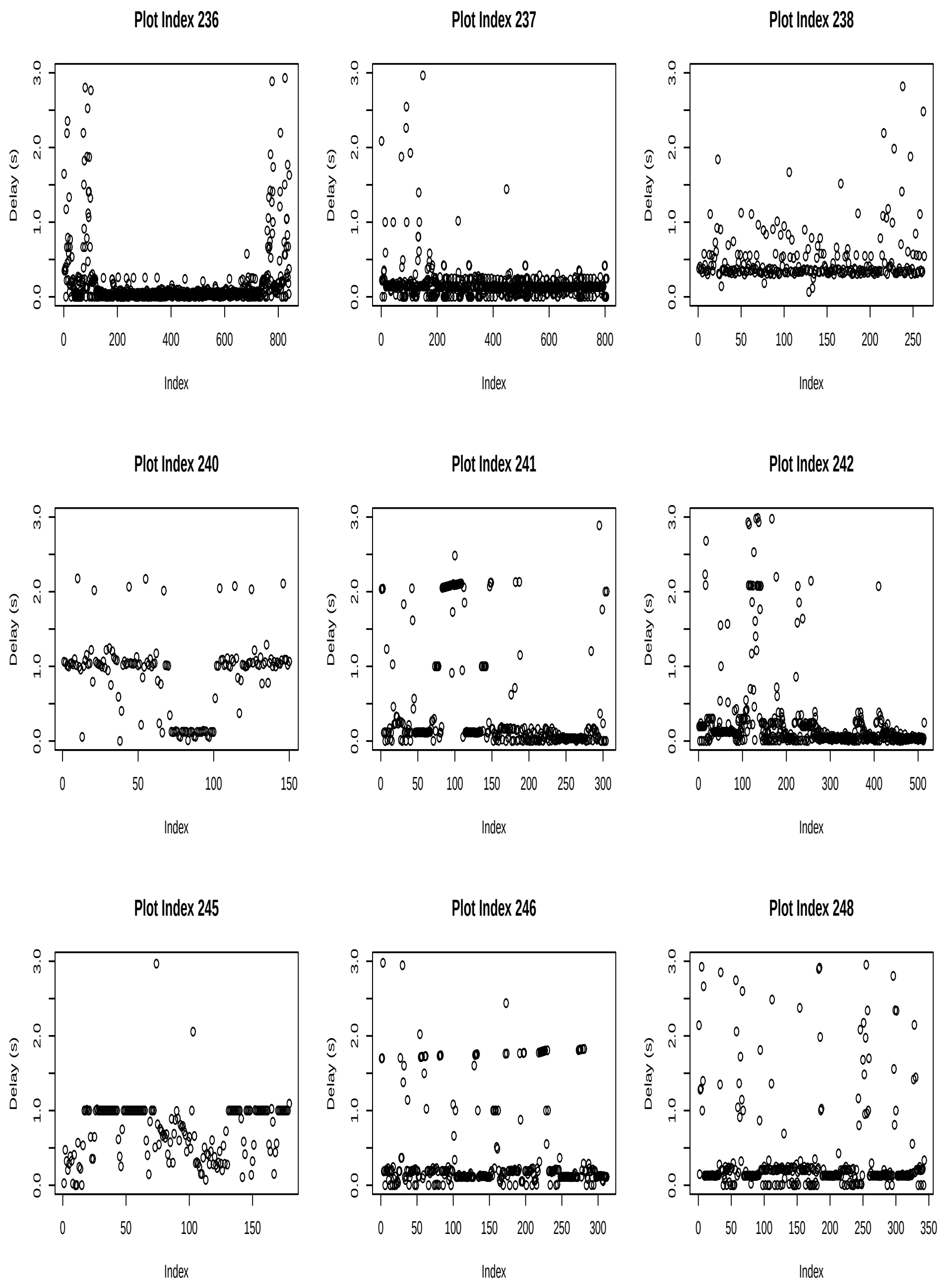


Figure B.17: Plot of Erroneous Port 22 Times removed from experiment

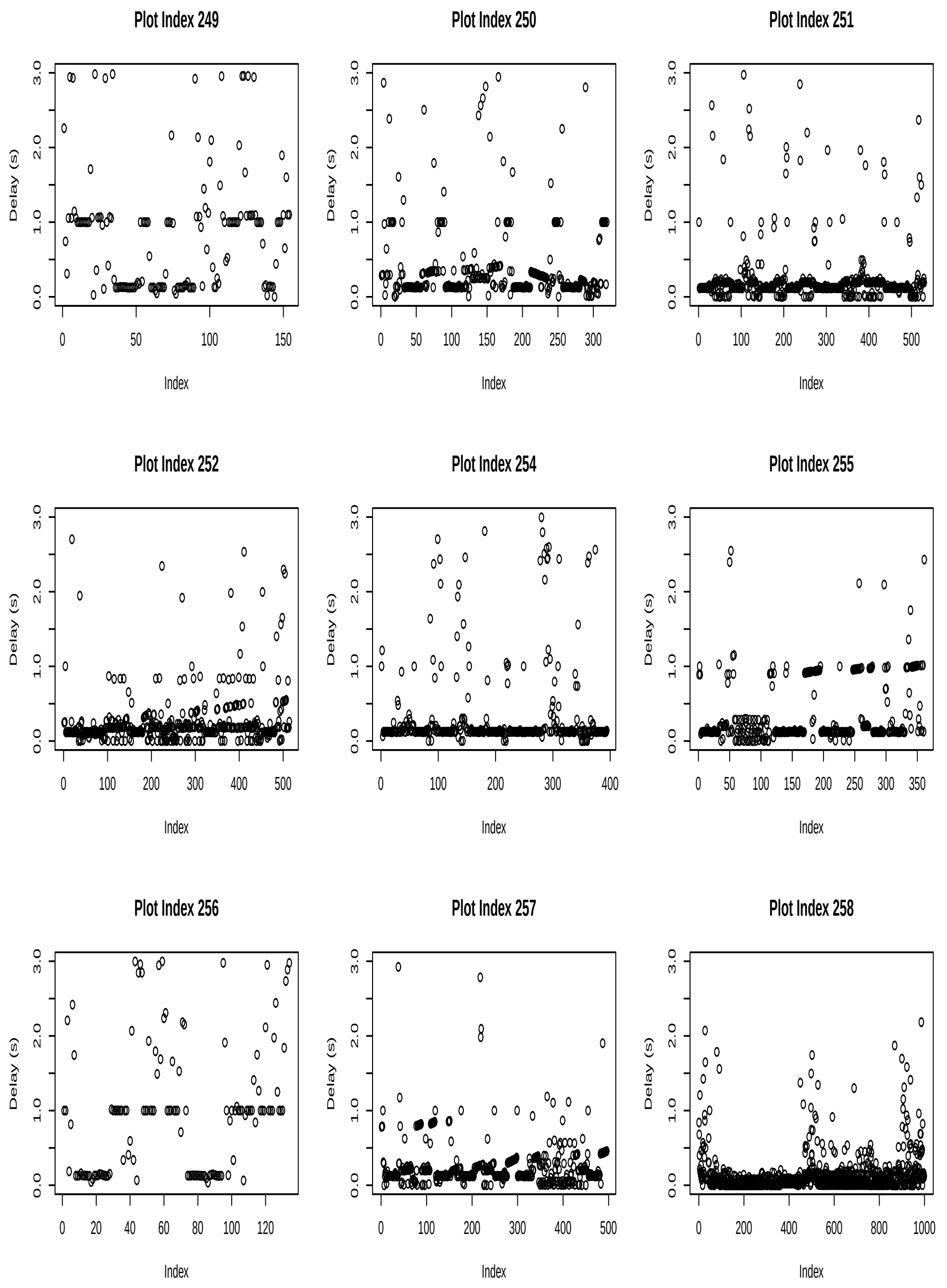


Figure B.18: Plot of Erroneous Port 22 Times removed from experiment

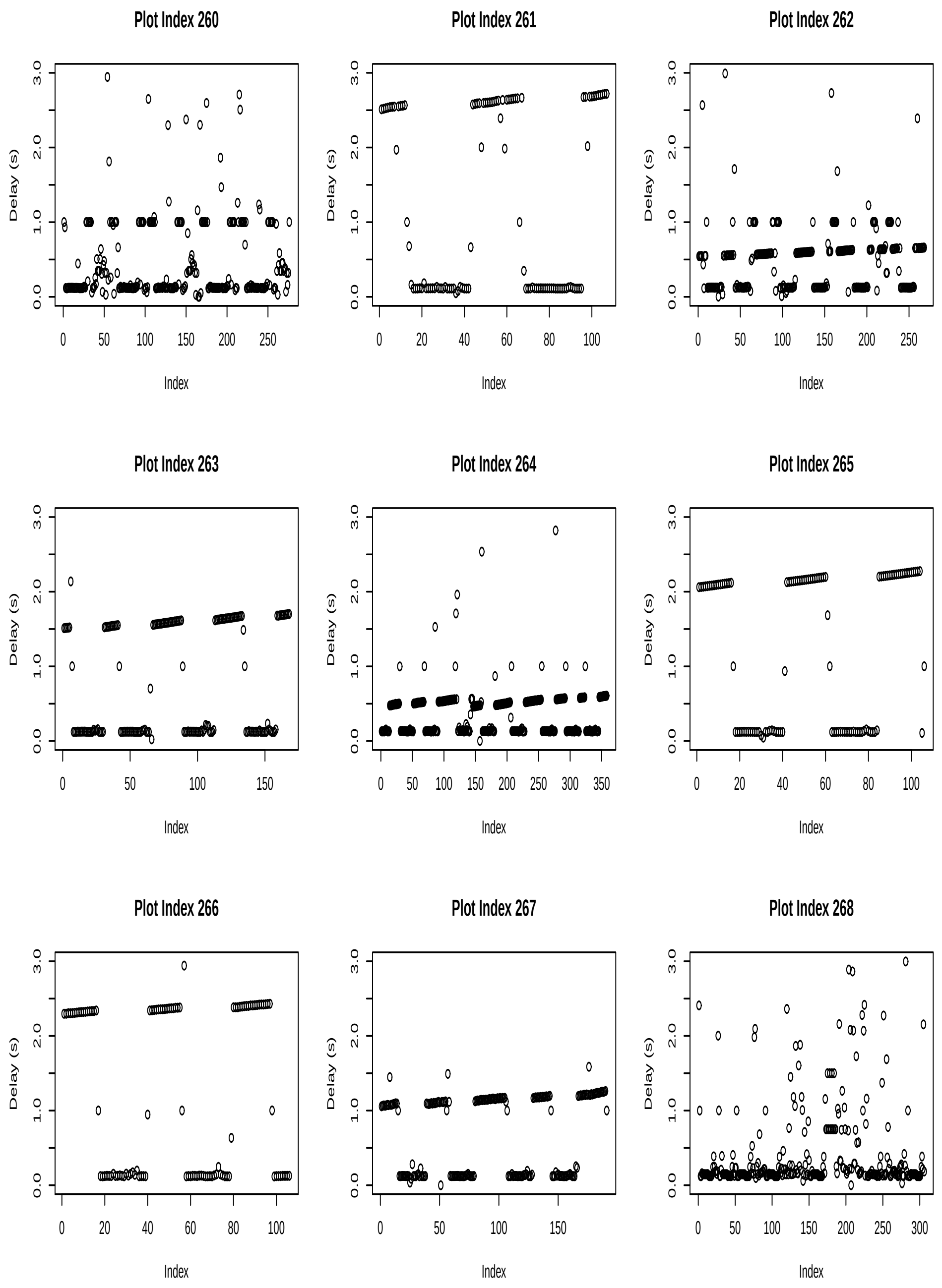


Figure B.19: Plot of Erroneous Port 22 Times removed from experiment

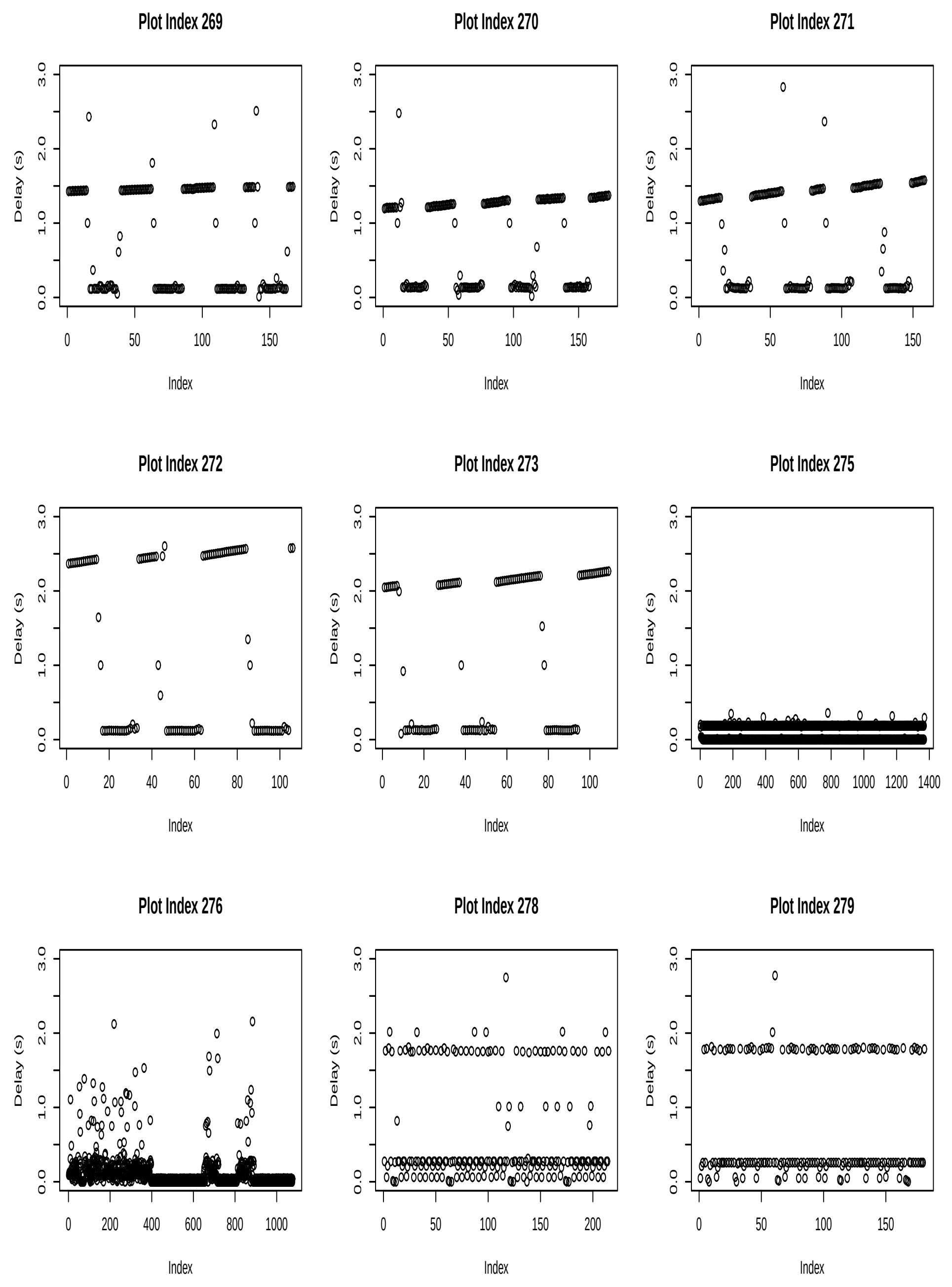


Figure B.20: Plot of Erroneous Port 22 Times removed from experiment

Bibliography

[1] Cheung, S., U. Lindqvist, and M.W. Fong. “Modeling multistep cyber attacks for scenario recognition”. *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, 284–292. IEEE, 2003.

[2] Coskun, B. and N. Memon. “Online Sketching of Network Flows for Real-Time Stepping-Stone Detection”. *Computer Security Applications Conference, 2009. ACSAC ’09. Annual*, 473 –483. dec. 2009. ISSN 1063-9527.

[3] Department of the Air Force, Air Force Material Command. “Network Attack Traceback”. Solicitation Number: Reference-Number-BAA-05-04-IFKA, April 2005. URL `https://www.fbo.gov/spg/USAF/AFMC/AFRLRRS/Reference-Number-BAA-05-04-IFKA/listing.html`.

[4] Divsalar, D., H. Jin, and R.J. McEliece. “Coding theorems for turbo-like codes”. *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, volume 36, 201–210. UNIVERSITY OF ILLINOIS, 1998.

[5] Donoho, David L., Ana Georgina Flesia, Umesh Shankar, Vern Paxson, Jason Coit, and Stuart Staniford. “Multiscale stepping-stone detection: detecting pairs of jittered interactive streams by exploiting maximum tolerable delay”. *Proceedings of the 5th international conference on Recent advances in intrusion detection*, RAID’02, 17–35. Springer-Verlag, Berlin, Heidelberg, 2002. ISBN 3-540-00020-8.

[6] Gourdin, B., C. Soman, H. Bojinov, and E. Bursztein. “Towards secure embedded web interfaces”. *Proceedings of the Usenix Security Symposium*. 2011.

[7] Heberlein, L.T., B. Mukherjee, and K.N. Levitt. “Internetwork Security Monitor: An Intrusion-Detection System for Large-Scale Networks”. *Proceedings of the National Computer Security Conference*, 262–271. DIANE Publishing Company, 1992.

[8] Houmansadr, Amir and Nikita Borisov. “SWIRL: A Scalable Watermark to Detect Correlated Network Flows.” *18th Annual Network and Distributed System Security Symposium (NDSS’11)*. 2011.

[9] Houmansadr, Amir and Nikita Borisov. “Towards Improving Network Flow Watermarks Using the Repeat-Accumulate Codes”. *36th International Conference on Acoustics, Speech and Signal Processing*. 2011.

[10] Houmansadr, Amir, Negar Kiyavash, and Nikita Borisov. “RAINBOW: A Robust And Invisible Non-Blind Watermark for Network Flows.” *16th Annual Network and Distributed System Security Symposium (NDSS’09)*. 2009.

[11] Jakobsson, Markus and Zulfikar Ramzan. *Crimeware: Understanding New Attacks and Defenses*. Addison-Wesley Professional, 2008. ISBN 0321501950.

[12] Jung, H. T. “Caller Identification System in the Internet Environment”. *Proceedings of the 4th Usenix Security Symposium, 1993*, 1993.

[13] Kiyavash, Negar, Amir Houmansadr, and Nikita Borisov. “Multi-flow attacks against network flow watermarking schemes”. *Proceedings of the 17th conference on Security symposium*, 307–320. USENIX Association, Berkeley, CA, USA, 2008.

[14] Olbrich, M., F. Nadolni, F. Idzikowski, and H. Woesner. “Measurements of Path Characteristics in PlanetLab”. 2009.

[15] Omar, M.N., L. Siregar, and R. Budiarto. “Hybrid stepping stone detection method”. *Distributed Framework and Applications, 2008. DFmA 2008. First International Conference on*, 134 –138. oct. 2008.

[16] Paxson, V. and S. Floyd. “Wide area traffic: the failure of Poisson modeling”. *IEEE/ACM Transactions on Networking (ToN)*, 3(3):226–244, 1995.

[17] SecureWorks. “Compromised US and Chinese Computers Launch Greatest Number of Cyber Attacks, according to SecureWorks’ Data”. Press Release, September 2008. `http://www.secureworks.com/media/press_releases/PR/13627/`.

[18] Singhal, A. and X. Ou. “Security Risk Analysis of Enterprise Networks Using Probabilistic Attack Graphs”. 2011.

[19] Snapp, S.R., J. Brentano, G.V. Dias, T.L. Goan, L.T. Heberlein, C.L. Ho, K.N. Levitt, B. Mukherjee, S.E. Smaha, T. Grance, et al. “DIDS (Distributed Intrusion Detection System)- Motivation, Architecture, and An Early Prototype”. *Internet besieged: countering cyberspace scofflaws*, 211–227, 1998.

[20] Staniford-Chen, S. and L.T. Heberlein. “Holding intruders accountable on the Internet”. *Security and Privacy, 1995. Proceedings., 1995 IEEE Symposium on*, 39 –49. may 1995.

[21] Sterne, Daniel F., Kelly Djahandari, Brett Wilson, Bill Babson, Dan Schnackenberg, Harley Holliday, and Travis Reid. “Autonomic Response to Distributed Denial of Service Attacks”. *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, RAID ’00, 134–149. Springer-Verlag, London, UK, 2001. ISBN 3-540-42702-3.

[22] Strayer, W., David Lapsely, Robert Walsh, and Carl Livadas. “Botnet Detection Based on Network Behavior”. Wenke Lee, Cliff Wang, and David Dagon (editors), *Botnet Detection*, volume 36 of *Advances in Information Security*, 1–24. Springer US, 2008. ISBN 978-0-387-68768-1.

[23] Strayer, W.T., R. Walsh, C. Livadas, and D. Lapsley. “Detecting Botnets with Tight Command and Control”. *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, 195 –202. nov. 2006. ISSN 0742-1303.

[24] Walsworth, Colby, Emile Aben, kc claffy, and Dan Andersen. "The CAIDA Anonymized 2009 Internet Traces - January - December". URL `http://www.caida.org/data/passive/passive_2009_dataset.xml`.

[25] Wang, Xinyuan, Shiping Chen, and Sushil Jajodia. "Network Flow Watermarking Attack on Low-Latency Anonymous Communication Systems". *Security and Privacy, IEEE Symposium on*, 0:116–130, 2007. ISSN 1081-6011.

[26] Wang, Xinyuan and D. Reeves. "Robust Correlation of Encrypted Attack Traffic through Stepping Stones by Flow Watermarking". *Dependable and Secure Computing, IEEE Transactions on*, 8(3):434 –449, may-june 2011. ISSN 1545-5971.

[27] Wang, Xinyuan, Douglas Reeves, and S. Wu. "Inter-Packet Delay Based Correlation for Tracing Encrypted Connections through Stepping Stones". Dieter Gollmann, Gnther Karjoth, and Michael Waidner (editors), *Computer Security ESORICS 2002*, volume 2502 of *Lecture Notes in Computer Science*, 244–263. Springer Berlin / Heidelberg, 2002.

[28] Wang, Xinyuan, Douglas S. Reeves, and S. Felix. "Tracing Based Active Intrusion Response". *In Journal of Information Warfare*, 2001. 2001.

[29] Wang, Xinyuan, Douglas S. Reeves, S. Felix Wu, and Jim Yuill. "Sleepy watermark tracing: an active network-based intrusion response framework". *Proceedings of the 16th international conference on Information security: Trusted information: the new decade challenge*, Sec '01, 369–384. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[30] Yoda, Kunikazu and Hiroaki Etoh. "Finding a Connection Chain for Tracing Intruders". Frdric Cuppens, Yves Deswarte, Dieter Gollmann, and Michael Waidner (editors), *Computer Security - ESORICS 2000*, volume 1895 of *Lecture Notes in Computer Science*, 191–205. Springer Berlin / Heidelberg, 2000.

[31] Yu, Wei, Xinwen Fu, Steve Graham, Dong Xuan, and Wei Zhao. "DSSS-Based Flow Marking Technique for Invisible Traceback". *Security and Privacy, IEEE Symposium on*, 0:18–32, 2007. ISSN 1081-6011.

[32] Zhang, Yin and Vern Paxson. "Detecting Stepping Stones". *Proceedings of the 9th USENIX Security Symposium*. Denver, CO, USA, August 2000.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704–0188*

1. REPORT DATE *(DD–MM–YYYY)* 22–03–2012	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Aug 2010 — Mar 2012
4. TITLE AND SUBTITLE Scalable Wavelet-Based Active Network Stepping Stone Detection		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Gilbert, Joseph I., MAJ USA		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way Wright-Patterson AFB OH 45433-7765		**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GE/ENG/12-17
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Center for Cyberspace Research (Raines, Richard A. Dr.) 2950 Hobson Way, Bldg 641 Wright Patterson AFB, OH 45433 ((937) 255-6565 x4278, richard.raines@afit.edu)		**10. SPONSOR/MONITOR'S ACRONYM(S)** AFIT/CCR
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Network intrusions leverage vulnerable hosts as stepping stones to penetrate deeper into a network and mask malicious actions from detection. This research focuses on a novel active watermark technique using Discrete Wavelet Transformations to mark and detect interactive network sessions. This technique is scalable, nearly invisible and resilient to multi-flow attacks. The watermark is simulated using extracted timestamps from the CAIDA 2009 dataset and replicated in a live environment. The simulation results demonstrate that the technique accurately detects the presence of a watermark at a 5% False Positive and False Negative rate for both the extracted timestamps as well as the empirical tcplib distribution. The watermark extraction accuracy is approximately 92%. The live experiment is implemented using the Amazon Elastic Compute Cloud. The client system sends marked and unmarked packets from California to Virginia using stepping stones in Tokyo, Ireland and Oregon. Five trials are conducted using simultaneous watermarked and unmarked samples. The live results are similar to the simulation and provide evidence demonstrating the effectiveness in a live environment to identify stepping stones.

15. SUBJECT TERMS

stepping stone, watermark, network, detection, DWT, wavelet

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Lt Col David Robinson, PhD
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U	UU	119	(937) 255–3636, x4598; david.robinson@afit.edu